

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN



BÁO CÁO TIỂU LUẬN

DỮ LIỆU LỚN

Đề tài:

**Phân tích dữ liệu sản phẩm bộ nhớ trên Amazon bằng thuật
toán phân cụm**

Giảng viên hướng dẫn:

ThS. Nguyễn Hồ Duy Trí

Sinh viên thực hiện:

Nguyễn Hoàng Đăng Khoa – 21520999

Cù Ngọc Hoàng - 21522086

Nguyễn Trần Gia Kiệt - 21522258

Bùi Đình Triệu - 21521576

Thành phố Hồ Chí Minh, tháng 12 năm 2024

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN



BÁO CÁO TIỂU LUẬN

DỮ LIỆU LỚN

Đề tài:

**Phân tích dữ liệu sản phẩm bộ nhớ trên Amazon bằng thuật
toán phân cụm**

Giảng viên hướng dẫn:

ThS. Nguyễn Hồ Duy Trí

Sinh viên thực hiện:

Nguyễn Hoàng Đăng Khoa – 21520999

Cù Ngọc Hoàng - 21522086

Nguyễn Trần Gia Kiệt - 21522258

Bùi Đình Triệu - 21521576

Thành phố Hồ Chí Minh, tháng 12 năm 2024

LỜI CẢM ƠN

Trước hết, nhóm chúng em xin gửi lời cảm ơn sâu sắc đến tập thể quý thầy cô trường Đại học Công nghệ Thông tin - Đại học Quốc gia TP.HCM và quý thầy cô khoa Hệ thống thông tin đã tạo điều kiện, giúp chúng em học tập và có được những kiến thức cơ bản làm tiền đề giúp chúng em hoàn thành được dự án này.

Đặc biệt, nhóm chúng em xin gửi lời cảm ơn chân thành và sâu sắc tới thầy Nguyễn Hồ Duy Trí (Giảng viên giảng dạy lý thuyết và thực hành môn DỮ LIỆU LỚN – IS405). Nhờ sự hướng dẫn tận tình và chu đáo của thầy, nhóm chúng em đã học hỏi được nhiều kinh nghiệm và hoàn thành thuận lợi, đúng tiến độ cho dự án của mình.

Ngoài ra, chúng em cũng gửi lời cảm ơn đến tập thể lớp IS405.P11 khoảng thời gian qua đã đồng hành cùng nhau. Cảm ơn sự đóng góp của tất cả các bạn cho những buổi học luôn sôi nổi, thú vị và dễ tiếp thu.

Trong quá trình thực hiện tiểu luận, nhóm chúng em luôn giữ một tinh thần cầu tiến, học hỏi và cải thiện từ những sai lầm, tham khảo từ nhiều nguồn tài liệu khác nhau và luôn mong đạt được kết quả nhất có thể. Tuy nhiên, do vốn kiến thức còn hạn chế trong quá trình trau dồi từng ngày, nhóm chúng em không thể tránh được những sai sót, vì vậy chúng em mong rằng quý thầy cô sẽ đưa ra nhận xét một cách chân thành để chúng em học hỏi thêm kinh nghiệm nhằm mục đích phục vụ tốt các dự án khác trong tương lai. Xin chân thành cảm ơn quý thầy cô!

Nhóm thực hiện

[illegible]

.....

.....

.....

.....

.....

.....

....., ngày.....tháng.....năm 2024

Người nhận xét

(Ký tên và ghi rõ họ tên)

MỤC LỤC

CHƯƠNG 1: TỔNG QUAN ĐỀ TÀI.....	12
1.1. Lý do chọn đề tài	12
1.2. Giới thiệu nguồn dữ liệu	12
1.3. Mô tả dữ liệu	13
1.4. Mô tả bài toán	15
CHƯƠNG 2: LẤY DỮ LIỆU TỪ AMAZON.....	17
2.1. Apache Airflow	17
2.1.1. Giới thiệu	17
2.1.2. Kiến trúc	17
2.2. Crawl data từ Amazon phục vụ cho đồ án.....	18
2.2.1. Sử dụng Docker để setup Airflow.....	18
2.2.2. Crawl Data từ Amazon.....	<i>Error! Bookmark not defined.</i>
CHƯƠNG 3: KỸ THUẬT TIỀN XỬ LÝ	26
3.1. Làm sạch dữ liệu.....	26
3.1.1. Loại bỏ cột bị thiếu dữ liệu	26
3.1.2. Xử lý cột price, old_price, rating	26
3.1.3. Xử lý cột reviews, purchases	28
3.1.4. Xử lý cột Brands	30

3.1.5.	<i>Xử lý cột Memory Storage Capacity, Digital Storage Capacity</i>	32
3.1.6.	<i>Xử lý cột Hardware Interface</i>	35
3.1.7.	<i>Xử lý cột Write Speed, Read Speed</i>	37
3.1.8.	<i>Xử lý cột Color</i>	39
3.1.9.	<i>Loại bỏ các hàng bị thiếu dữ liệu và bị lặp</i>	42
3.2.	Chuẩn bị dữ liệu cho kỹ thuật khai thác	42
3.2.1.	<i>Kỹ thuật MinMaxScaler:</i>	42
3.2.2.	<i>Kỹ thuật StringIndex + One-hot Encoding:</i>	43
CHƯƠNG 4: KỸ THUẬT KHAI THÁC DỮ LIỆU		45
4.1.	K-Means	45
4.1.1.	<i>Cơ sở lý thuyết</i>	45
4.1.2.	<i>Thực nghiệm</i>	47
4.2.	DBScan	53
4.2.1.	<i>Cơ sở lý thuyết</i>	53
4.2.2.	<i>Thực nghiệm</i>	58
CHƯƠNG 5: KẾT QUẢ ĐẠT ĐƯỢC		66
5.1.	Kết quả	66
5.1.1.	<i>K-Means</i>	66
5.1.2.	<i>DBScan</i>	67
5.2.	So sánh, đánh giá	68

5.2.1. K-Means.....	68
5.2.2. DBScan	72
CHƯƠNG 6: KẾT LUẬN	75
6.1. Ưu điểm	75
6.2. Hạn chế.....	75
6.3. Hướng phát triển	75
PHÂN CÔNG CÔNG VIỆC	77
TÀI LIỆU THAM KHẢO	79

DANH MỤC HÌNH ẢNH

Figure 1. Trang Amazon	13
Figure 2. File Excel dữ liệu	13
Figure 3. Hiển thị dưới dạng Dataframe	15
Figure 4. Biểu đồ phần trăm dữ liệu bị thiếu mỗi cột	15
Figure 5 Xây dựng Image từ Docker File	18
Figure 6 Cấu hình Docker Compose	19
Figure 7 Hàm giả lập User-Agent để tránh bị phát hiện là bot	19
Figure 8 Hàm Random thời gian chống bot	20
Figure 9 Hàm lấy tiêu đề sản phẩm	20
Figure 10 Hàm lấy giá hiện tại của sản phẩm	20
Figure 11 Hàm lấy giá gốc của sản phẩm	20
Figure 12 Lấy URL sản phẩm	21
Figure 13 Lấy đánh giá sản phẩm	21
Figure 14 Hàm lấy số lượng đánh giá	21
Figure 15 Hàm số lượng bán ra	21
Figure 16 Một phần của hàm lấy thông tin chi tiết sản phẩm	22
Figure 17 Một phần của hàm xử lý các tác vụ chính	22
Figure 18 Các tham số DAG	23
Figure 19 Các tham số gọi hàm thực hiện	23
Figure 20 Chia nhiều file thực hiện	24
Figure 21 Kết quả file CSV	24
Figure 22 Kết quả file json	24
Figure 23 Hàm merge file csv (1)	25
Figure 24 Hàm merge file csv (2)	25
Figure 25. Các cột có thể được giữ lại	26
Figure 26. Đổi tên cột	26

Figure 27. Xử lý cột price, old_price và rating	27
Figure 28. Kết quả price, old_price và rating	27
Figure 29. Xử lý cột reviews, purchases	28
Figure 30. Kết quả reviews, purchases	29
Figure 31. Đếm số Brands.....	30
Figure 32. Lọc Brands.....	31
Figure 33. Xử lý cột Brands	31
Figure 34. Kết quả Brands	32
Figure 35. Xử lý cột Memory Storage Capacity, Digital Storage Capacity (1).....	33
Figure 36. Xử lý cột Memory Storage Capacity, Digital Storage Capacity (2).....	33
Figure 37. Xử lý cột Memory Storage Capacity, Digital Storage Capacity (3).....	34
Figure 38. Xử lý cột Memory Storage Capacity, Digital Storage Capacity (4).....	34
Figure 39. Kết quả Memory Storage Capacity, Digital Storage Capacity	35
Figure 40. Xử lý cột Hardware Interface	36
Figure 41. Kết quả Hardware Interface.....	37
Figure 42. Xử lý cột Write Speed	38
Figure 43. Kết quả Write Speed.....	38
Figure 44. Xử lý cột Read Speed	39
Figure 45. Kết quả Read Speed.....	39
Figure 46. Đếm số Color.....	40
Figure 47. Xử lý cột Color	41
Figure 48. Kết quả Color.....	41
Figure 49. Loại bỏ hàng bị lặp và bị thiếu dữ liệu	42
Figure 50. Kết quả dataframe sau làm sạch dữ liệu	42
Figure 51. Kỹ thuật MinMaxScaler lên các cột dữ liệu số	43
Figure 52. StringIndexer + One-hot Encoding.....	44
Figure 53 Điều kiện dừng	47
Figure 54. Eps-neighborhood.....	54

Figure 55. Directly density-reachable	55
Figure 56. Density-reachable	56
Figure 57. Density-connected	56
Figure 58. Phân loại các điểm trong DBScan	57
Figure 59. Mô tả song song hóa giải thuật DBScan.....	60
Figure 60. Xử lý dữ liệu đầu vào	61
Figure 61. Tính khoảng cách Euclidean giữa các điểm	61
Figure 62. Khởi tạo eps và minPts và broadcast giá trị	62
Figure 63. Tìm hàng xóm cho tất cả các điểm	62
Figure 64. Phân loại điểm core và khởi tạo cluster ban đầu cho các điểm	62
Figure 65. Tìm các điểm noise.....	63
Figure 66. Tìm các điểm core	63
Figure 67. Tìm các điểm border.....	63
Figure 68. Cập nhật các điểm border vào dataframe	63
Figure 69. Mở rộng cụm từ một điểm core	64
Figure 70. Hàm mở rộng cụm	65
Figure 71. Số lượng dữ liệu ở mỗi cụm	66
Figure 72. Kết quả phân cụm bằng DBScan	67
Figure 73 Chỉ số silhouette score cài đặt không dùng thư viện máy học	69
Figure 74 Chỉ số silhouette score cài đặt dùng thư viện máy học	70
Figure 75 Phân cụm không dùng thư viện máy học	71
Figure 76 Phân cụm sử dụng thư viện máy học.....	72
Figure 77. Chỉ số Silhoutte của DBScan.....	73
Figure 78. Heatmap của DBScan	73

CHƯƠNG 1: TỔNG QUAN ĐỀ TÀI

1.1. Lý do chọn đề tài

- Trong thời đại thương mại điện tử bùng nổ, Amazon đã trở thành một trong những nền tảng mua sắm trực tuyến lớn nhất toàn cầu với hàng triệu sản phẩm thuộc nhiều danh mục khác nhau. Điều này đặt ra thách thức lớn cho cả người tiêu dùng và doanh nghiệp trong việc lựa chọn sản phẩm, tối ưu hóa chiến lược kinh doanh, và phân tích dữ liệu trên quy mô lớn.
- Việc phân cụm các loại sản phẩm trên Amazon giúp cung cấp một cái nhìn tổng quan về cơ cấu nhóm sản phẩm và mang lại nhiều lợi ích thiết thực như:
 - Cá nhân hóa trải nghiệm người dùng.
 - Hỗ trợ doanh nghiệp tối ưu chiến lược marketing
 - Quản lý kho hàng hiệu quả.
 - Khám phá xu hướng thị trường và nhóm sản phẩm tiềm năng.
- Với những lợi ích trên, nhóm quyết định sử dụng thuật toán phân cụm K-Means và DBSCAN để phân loại các sản phẩm bộ nhớ dựa trên bộ dữ liệu thu thập từ Amazon, phục vụ mục tiêu học tập và nghiên cứu.

1.2. Giới thiệu nguồn dữ liệu

- Nguồn dữ liệu: <https://www.amazon.com/s?i=computers-intl-ship&bbn=16225007011&rh=n%3A16225007011%2Cn%3A1292110011>



- Nguồn dữ liệu được thu nhập từ trang amazon về các sản phẩm bộ nhớ bao gồm có 3740 hàng và 140 cột dữ liệu.
- Mỗi hàng dữ liệu sẽ bao gồm thông tin của sản phẩm bộ nhớ, được chia thành 2 loại chủ yếu là USB và SSD + HDD. Các thông tin bao gồm, giá cả, số điểm đánh giá, lượt đánh giá, thông số kỹ thuật, etc...

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
		title	price	old_price	product_rating	reviews	purchases_detail	brand	detail_me	detail_har	detail_spc	detail_cor	detail_dig	detail_har	detail_cor	detail_ins	detail_red	detail_wri	detail_sty	detail_har	detail_col	detail_desc	detail_har	
1	2	MFI Certif	\$39.99		https://www.amazo	96,500+	boug	SUDEHO	256 GB	USB, Light	Lightweig	USB												
3	4	fanxiang 2	\$36.99	\$39.99	https://www.amazo	204	List:	fanxiang			Dual USB 3.0	256 GB	USB 3.0	Solid Stati	Laptop, Di	External H	350 Megal	90 Megab	Per Second					
5	6	ThePhoto	\$92.96	\$149.99	https://www.amazo	2,182	3K+ boug	ThePhoto	256 GB	USB	Light Weight													
7	8	Seagate P	\$69.99	\$79.99	https://wv.4.6 out of	261,799	10K+ boug	Seagate			Portable USB	2 TB	USB 3.0	Mechanic	Laptop, Di	Plug In				Portable F	3.5 Inches	Black	Easy store and ace	
9	10	SAMSUNG	\$169.86	\$249.99	https://www.amazo	10,202	10K+ boug	SAMSUNG			Backward PCIe Gen	2 TB	SATA 6 GB	Solid Stati	PC, Internal	Hard Drive				2.5 Inches	Informati	Reach max performa		
11	12	Seagate S	\$199.99	\$359.99	https://wv.4.6 out of	23,493	10K+ boug	SAMSUNG			Portable USB	2000 GB	NVMe	Solid Stati	Gaming C	Internal Hard Drive				Expansion 2.5	Inches	Black	Instantly expand the	
13	14	SAMSUNG	\$169.00	\$198.75	https://www.amazo	39,392	10K+ boug	SAMSUNG			Backward PCIe Gen	2 TB	NVMe	Solid Stati	Laptop, PC	Internal Hard Drive				980 PRO	2280	Inch	Black	Unleash the power c
15	16	WD_BLACK	\$173.99	\$199.99	https://wv.4.7 out of	70,255	10K+ boug	SAMSUNG			Backward PCIe Gen	1 TB	SATA 6 GB	Solid Stati	Laptop, PC	Internal Hard Drive				980 PRO	2280	Inch	Black	Fast speeds, Power Y
17	18	SanDisk A	\$259.99	\$359.99	https://wv.4.6 out of	30,925	9K+ boug	SanDisk			Hardware USB	4 TB	USB 3.2	Solid Stati	Tablet								Black	Your life 4 TB
19	20	WD_BLACK	\$149.99	\$189.99	https://www.amazo	22,658	10K+ boug	WD_BLACK			WD_BLACK NVMe	2000 GB	PCI E 3.2	Solid Stati	Laptop, M	Solid State Drive				SSD	2.5 Centin	Black	Strap in for breakne	
21	22	SAMSUNG	\$271.86	\$499.99	https://wv.4.7 out of	13,718	7K+ boug	SAMSUNG			Portable, USB 3.2	64 G4 TB	USB 3.2	Solid Stati	PC, Gamir	External Hard Drive					3.5 Inches	Black	Samsung T7 Shield P	
23	24	Western E	\$113.99	\$159.99	https://www.amazo	19,165	9K+ boug	WD_BLACK			WD_BLACK NVMe	2000 GB	Solid Stati	Solid Stati	Laptop, M	Solid State Drive				M.2 2280	f2	Inches	Black	Get ready to game e
25	26	JSL JDCTL	\$89.99	\$109.99	https://wv.4.3 out of	58	Typical:	JSL JDCTL	256 GB	USB	Auto Photo & Video Backup									158.48931	360°	Rotation		
27	28	1TB Extern	\$79.99		https://wv.4.6 out of	160	50+ boug	Fikot			Dust Resis USB	1 TB	Solid Stati	Solid Stati	Laptop, Gi	External Hard Drive					3.35	Inches	Space Gray	
29	30	MOSDART	\$9.99	\$13.99	https://wv.4.6 out of	13,954	400+ boug	MOSDART	64 GB	USB 2.0	Keychain lock, Capless Design,													
31	32	WD_BLACK	\$179.99	\$229.99	https://www.amazo	5,423	7K+ boug	WD_BLACK			Officially NVMe	2 TB	NVMe	Solid Stati	Gaming C	Internal Hard Drive				Up to 7,300MB/s		Black	Officially 2 TB	
33	34	SAMSUNG	\$278.14		https://www.amazo	33,496	5K+ boug	SAMSUNG			Portable, USB 3.2	64 G4 TB	USB 3.0	Solid Stati	PC, Gamir	External Hard Drive								
35	36	Crucial P3	\$237.99	\$359.99	https://www.amazo	17,538	7K+ boug	Crucial			Compact NVMe	2 TB	PCI E 4.0	Solid Stati	This drive	Internal Hard Drive				P3 Plus				Create, Explore, Stor
37	38	Crucial BX	\$89.99	\$79.99	https://www.amazo	6,192	10K+ boug	ThePhoto	128 GB	USB	Lightweight, Dual USB Connector										Up to 20 MB/s			Valuable 4 TB
39	40	Crucial BX	\$58.49	\$94.99	https://wv.4.6 out of	121,636	10K+ boug	Crucial			Data Reco SATA	1 TB	Solid Stati	Solid Stati	Laptop, Di	Internal Hard Drive								
41	42	Amslon R	\$15.67		https://wv.4.7 out of	19,747	10K+ boug	Amslon R	128 GB	USB	Lightweight										130 Megab	Per Second		

Figure 2. File Excel dữ liệu

Thuộc tính	Mô tả
title	Tiêu đề sản phẩm

price	Giá hiện tại của sản phẩm
old_price	Giá gốc của sản phẩm (nếu có)
product_url	Liên kết đến trang chi tiết sản phẩm
rating	Đánh giá trung bình của sản phẩm
reviews	Số lượng đánh giá
purchases	Số lượng đã mua
detail_brand	Thương hiệu sản phẩm
detail_memory_storage_capacity	Dung lượng lưu trữ
detail_hardware_interface	Giao diện phần cứng
detail_special_feature	Tính năng đặc biệt
detail_read_speed	Tốc độ đọc dữ liệu
detail_write_speed	Tốc độ ghi dữ liệu
detail_item_dimensions	Kích thước sản phẩm
detail_weight	Trọng lượng sản phẩm
detail_operating_system	Hệ điều hành tương thích
detail_specific_uses_for_product	Mục đích sử dụng cụ thể
detail_color	Màu sắc của sản phẩm
detail_model_name	Tên model của sản phẩm
detail_material	Vật liệu của sản phẩm
detail_included_components	Các thành phần đi kèm
detail_compatible_devices	Các thiết bị tương thích
detail_data_transfer_rate	Tốc độ truyền dữ liệu
detail_power_source	Nguồn năng lượng

Đánh giá sơ bộ:

title	price old_price	product_url	rating reviews	purchases	detail_brand detail_memory_storage_capacity detail_hardware_interface detail_special_feature detail_connectivity_technology
MPi Certified 256...	\$39.99	NULL	NULL	96 500+ bought in pa...	NULL
FanLang 256GB F1...	\$36.99	NULL	208	List:	fanLang
ThePhotoStick® On...	\$92.96	NULL	2,182 3K+ bought in pas...	ThePhotoStick	256 GB
Seagate Portable ...	\$69.99	NULL	14.6 out of 5 stars 263,790 18K+ bought in pa...	Seagate	256 GB
SAMSUNG 990 PRO S...	\$249.99	NULL	18,262 18K+ bought in pa...	SAMSUNG	1TB
Seagate Storage E...	\$199.99	NULL	14.6 out of 5 stars 23,493 18K+ bought in pa...	Seagate	1TB
SAMSUNG 990 PRO S...	\$169.00	NULL	14.7 out of 5 stars 39,332 18K+ bought in pa...	SAMSUNG	1TB
SAMSUNG 990 EVO S...	\$73.00	NULL	14.7 out of 5 stars 1,591 18K+ bought in pa...	SAMSUNG	1TB
SanDisk 4TB Extre...	\$359.99	NULL	14.6 out of 5 stars 70,295 9K+ bought in pas...	SanDisk	4TB
WD_BLACK 2TB SN85...	\$149.99	NULL	14.6 out of 5 stars 22,658 18K+ bought in pa...	WD_BLACK	2TB
SAMSUNG T7 Shield...	\$271.86	NULL	14.7 out of 5 stars 37,718 7K+ bought in pas...	SAMSUNG	1TB
Western Digital M...	\$119.99	NULL	15,165 9K+ bought in pas...	WD_BLACK	1TB
PSL 3D10C 256GB 3...	\$89.99	NULL	4.3 out of 5 stars 58	Typical:	PSL 3D10C
1TB External SSD...	\$79.99	NULL	168 50+ bought in pas...	Filmora	1TB
WD_BLACK 4TB SN85...	\$9.99	NULL	14.6 out of 5 stars 13,954 400+ bought in pa...	WD_BLACK	4TB
WD_BLACK 2TB SN85...	\$179.99	NULL	5,423 7K+ bought in pas...	WD_BLACK	2TB
SAMSUNG T7 Portab...	\$278.14	NULL	33,496 5K+ bought in pas...	SAMSUNG	1TB

Figure 3. Hiển thị dưới dạng Dataframe

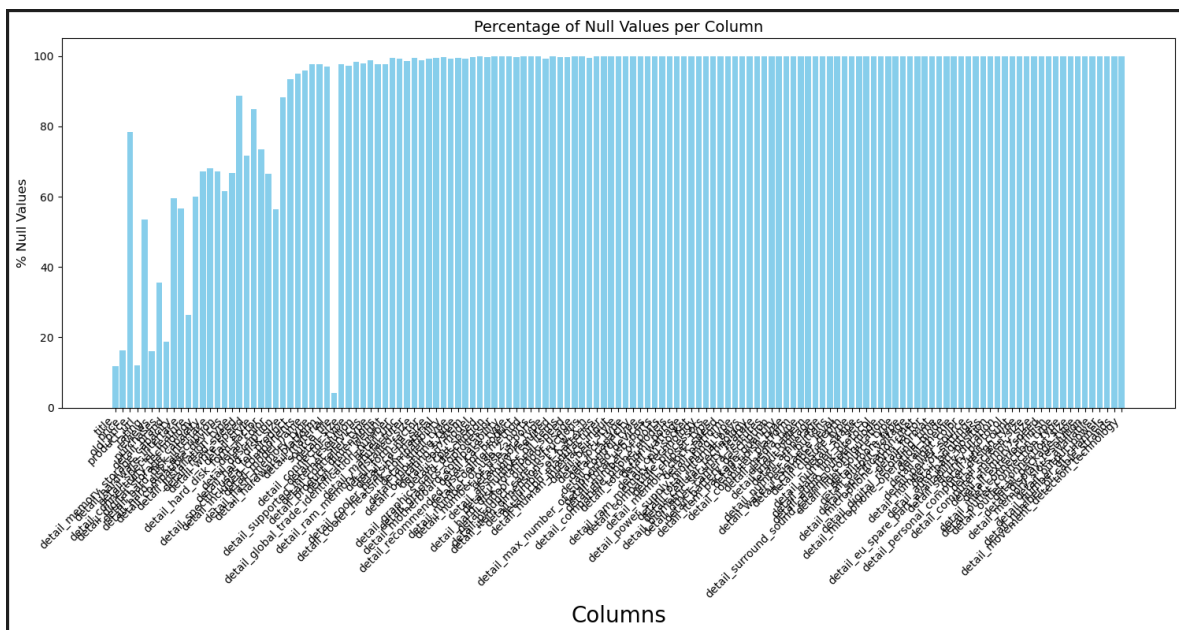


Figure 4. Biểu đồ phần trăm dữ liệu bị thiếu mỗi cột

- Biểu đồ trên cho ta thấy lượng lớn dữ liệu bị thiếu, nhiều cột gần như không có dữ liệu.
- Cột title qua đánh giá sơ bộ có thể chứa dữ liệu liên quan đến thông số của sản phẩm, có thể khai thác trong quá trình làm sạch dữ liệu để có thể dữ lại một số cột.
- Do dữ liệu bị thiếu quá nhiều nên chỉ có thể tập trung xử lý trước vấn đề bị thiếu do không có có dữ liệu đủ để khai thác.

1.4. Mô tả bài toán

- Trong báo cáo này, nhóm thực hiện phương pháp phân cụm bằng thuật toán

KMeans và DBScan dựa trên sự tương đồng của dữ liệu và tìm ra tập luật để dự đoán. Từ đó giúp rõ hơn các sản phẩm bộ nhớ được phân phối và bán trên sàn thương mại điện tử Amazon.

CHƯƠNG 2: LẤY DỮ LIỆU TỪ AMAZON

2.1. Apache Airflow

2.1.1. Giới thiệu

- Apache Airflow là một nền tảng mã nguồn mở dùng để phát triển, lên lịch và giám sát các luồng công việc theo lô (batch-oriented workflows). Khung mở rộng bằng Python của Airflow cho phép bạn xây dựng các luồng công việc kết nối với hầu như bất kỳ công nghệ nào. Giao diện web hỗ trợ quản lý trạng thái của các luồng công việc. Airflow có thể triển khai theo nhiều cách, từ một tiến trình duy nhất trên máy tính cá nhân của bạn đến một cấu hình phân tán để hỗ trợ ngay cả các luồng công việc lớn nhất.

2.1.2. Kiến trúc

- Kiến trúc của Airflow bao gồm nhiều thành phần. Các phần sau đây mô tả chức năng của từng thành phần và liệu chúng có cần thiết cho việc cài đặt Airflow ở mức tối thiểu hay thành phần tùy chọn để đạt được khả năng mở rộng, hiệu suất và khả năng mở rộng Airflow tốt hơn
- **Các thành phần cần thiết trong Airflow:** Một cài đặt tối thiểu của Airflow cần bao gồm các thành phần sau:

2.1.2.a. Scheduler

Nhiệm vụ:

Lên lịch và kích hoạt các luồng công việc (workflows) theo thời gian định trước.

Gửi các task đến executor để thực thi.

Executor:

Là một thuộc tính cấu hình của Scheduler, không phải là một thành phần tách biệt.

Chạy trong cùng tiến trình với Scheduler.

Airflow cung cấp nhiều loại executor có sẵn, hoặc bạn có thể tự viết

executor của riêng mình.

2.1.2.b. Webserver

Cung cấp giao diện người dùng để kiểm tra, kích hoạt và gỡ lỗi DAGs (Directed Acyclic Graphs) và các task.

2.1.2.c. Thư mục DAG Files

Chứa các file định nghĩa DAG.

Scheduler đọc các file này để xác định task cần chạy và thời điểm thực thi.

2.1.2.d. Metadata Database

Lưu trữ trạng thái và thông tin lịch sử của workflows và tasks.

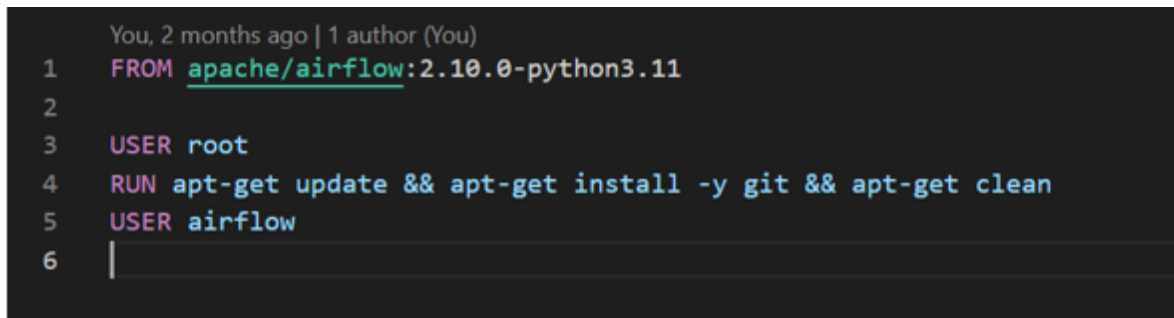
Cơ sở dữ liệu này là thành phần bắt buộc để Airflow hoạt động.

Hướng dẫn cài đặt Metadata Database có thể được tham khảo trong tài liệu Set up a Database Backend.

2.2. Crawl data từ Amazon phục vụ cho đồ án

2.2.1. Sử dụng Docker để setup Airflow

- Chọn phiên bản Airflow phù hợp với python để sử dụng trong Docker

A screenshot of a Dockerfile with a dark background and light-colored text. The text is as follows:

```
You, 2 months ago | 1 author (You)
1 FROM apache/airflow:2.10.0-python3.11
2
3 USER root
4 RUN apt-get update && apt-get install -y git && apt-get clean
5 USER airflow
6 |
```

Figure 5 Xây dựng Image từ Docker File

- Sử dụng docker compose để chạy container làm server cho airflow

```

5  services:
6
7      sleek-airflow:
8
9          image: airflow:latest
10
11
12
13      volumes:
14
15          - ./airflow:/opt/airflow
16
17
18
19      ports:
20
21          - "8080:8080"
22
23
24
25      command: airflow standalone
  
```

Figure 6 Cấu hình Docker Compose

1.1.1. Crawl Data từ Amazon

- Tạo tiêu đề HTTP (headers) ngẫu nhiên, bao gồm các User-Agent khác nhau, nhằm giả lập các trình duyệt khác nhau khi gửi request.

```

18 def generate_headers():
19     user_agents = [
20         'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.0.0',
21         'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.0.0',
22         'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.0.0',
23         'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.0.0',
24         'Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/119.0',
25         'Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:108.0) Gecko/20100101 Firefox/118.0',
26         'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/120.0.0.0',
27         'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/119.0.0.0',
28         'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.0.0',
29         'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.0.0',
30         'Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.0.0 Safari/537.36',
31         'Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.0.0 Safari/537.36'
32     ]
33     return {
34         'User-Agent': random.choice(user_agents),
35         'Accept-Language': 'en-US, en;q=0.5',
36         'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8',
37         'Referer': 'https://www.amazon.com/'
38     }
  
```

Figure 7 Hàm giả lập User-Agent để tránh bị phát hiện là bot

- Tạo một khoảng thời gian trễ ngẫu nhiên từ 5 đến 7 giây để tránh bị phát hiện là bot khi gửi nhiều request đến Amazon.

```
def get_random_delay():  
    return random.uniform(5, 7)
```

Figure 8 Hàm Random thời gian chống bot

- Lấy tiêu đề sản phẩm từ HTML sử dụng BeautifulSoup. Tìm các phần tử phù hợp dựa trên class CSS.

```
def get_price(soup):  
    try:  
        return soup.find('span', class_='a-price').find('span', class_='a-offscreen').text.strip()  
    except AttributeError:  
        return ""
```

Figure 9 Hàm lấy tiêu đề sản phẩm

- Lấy giá hiện tại của sản phẩm.

```
def get_title(soup):  
    try:  
        title_element = soup.find("a", class_="a-link-normal s-line-clamp-4 s-link-style a-text-normal")  
        if title_element:  
            h2_element = title_element.find("h2", class_="a-size-base-plus a-spacing-none a-color-base a-text-normal")  
            if h2_element:  
                span_element = h2_element.find("span")  
                return span_element.get_text(strip=True) if span_element else ""  
    except AttributeError:  
        return ""
```

Figure 10 Hàm lấy giá hiện tại của sản phẩm

- Lấy giá gốc của sản phẩm nếu có giảm giá.

```
def get_old_price(soup):  
    try:  
        return soup.find("div", class_="a-section aok-inline-block").find("span", class_="a-offscreen").text.strip()  
    except AttributeError:  
        return ""
```

Figure 11 Hàm lấy giá gốc của sản phẩm

- Tạo URL đầy đủ dẫn đến sản phẩm dựa trên thuộc tính href của thẻ HTML.

```
def get_product_url(soup):  
    try:  
        return "https://www.amazon.com" + soup.find("a", class_="a-link-normal s-line-clamp-4 s-link-style a-text-normal")['href']  
    except (TypeError, AttributeError):  
        return ""
```

Figure 12 Lấy URL sản phẩm

- Lấy đánh giá của sản phẩm

```
def get_product_url(soup):  
    try:  
        return "https://www.amazon.com" + soup.find("a", class_="a-link-normal s-line-clamp-4 s-link-style a-text-normal")['href']  
    except (TypeError, AttributeError):  
        return ""
```

Figure 13 Lấy đánh giá sản phẩm

- Lấy số lượng đánh giá cho sản phẩm.

```
def get_review_count(soup):  
    try:  
        return soup.find('span', class_='a-size-base s-underline-text').text.strip()  
    except AttributeError:  
        return ""
```

Figure 14 Hàm lấy số lượng đánh giá

- Lấy số lượng mua hàng nếu có thông tin này.

```
def get_purchase_count(soup):  
    try:  
        return soup.find("span", class_="a-size-base a-color-secondary").get_text(strip=True)  
    except AttributeError:  
        return ""
```

Figure 15 Hàm số lượng bán ra

- Lấy thêm thông tin chi tiết từ trang sản phẩm như mô tả, thương hiệu, kiểu dáng (style), và các đặc điểm kỹ thuật khác.

```
def get_product_details(product_url):
    try:
        headers = generate_headers()

        with requests.Session() as session:
            page = session.get(product_url, headers=headers, timeout=10)

            soup = BeautifulSoup(page.content, "html.parser")

            details = {}
            info_sections = [
                soup.find("div", class_="a-section a-spacing-small a-spacing-top-small"),
                soup.find("div", id="productDetails_techSpec_section_1")
            ]

            You, last week • update crawl

            style_section = soup.find("div", id="variation_style_name")
            if style_section:
                try:
                    style_label = style_section.find("label", class_="a-form-label")
```

Figure 16 Một phần của hàm lấy thông tin chi tiết sản phẩm

- Hàm chính thực hiện các tác vụ: khởi tạo thư mục để lưu trữ dữ liệu, lặp qua các trang, thu thập dữ liệu sản phẩm bằng các hàm, ghi dữ liệu thành các file json và csv.

```
def scrape_amazon_products_task(**kwargs):
    """
    Air-flow task to scrape Amazon products without pandas
    """
    base_url = kwargs.get('base_url', "https://www.amazon.com/s?i=computers-intl-ship&bnn=16225007011&rh=n%3A16225007011%2Cn%3A1292110011")
    max_pages = kwargs.get('max_pages', 50)
    start_page = kwargs.get('start_page', 70)
    fetch_details = kwargs.get('fetch_details', True)

    # Create session
    session = requests.Session()

    # Initialize list to store data
    products_data = []

    # Create output directory
    output_dir = "/opt/airflow/data/amazon_scraper_output"
    os.makedirs(output_dir, exist_ok=True)

    # Loop through pages
    for page in range(start_page, start_page + max_pages):
```

Figure 17 Một phần của hàm xử lý các tác vụ chính

- Định nghĩa các tham số mặc định cho DAG

```
default_args = {
    'owner': 'airflow',
    'depends_on_past': False,
    'start_date': days_ago(1),
    'email_on_failure': False,
    'email_on_retry': False,
    'retries': 1,
    'retry_delay': timedelta(minutes=5),
}
```

Figure 18 Các tham số DAG

- Tên dag, mô tả, lịch trình, nhiệm vụ gọi hàm chính thực hiện thu thập dữ liệu

```
with DAG(
    'amazon_product_scraper',
    default_args=default_args,
    description='Scrape Amazon product listings',
    schedule_interval=timedelta(days=7), # Run weekly
    catchup=False
) as dag:

    scrape_products = PythonOperator(
        task_id='scrape_amazon_products',
        python_callable=scrape_amazon_products_task,
        op_kwargs={
            'base_url': "https://www.amazon.com/s?i=computers-intl-ship&bbn=16225007011&rh=n%3A16225007011%2Cn%3A1292110011",
            'max_pages': 20,
            'start_page': 1,
            'fetch_details': True
        },
        dag=dag,
    )

scrape_products
```

Figure 19 Các tham số gọi hàm thực hiện

- Chia thành nhiều file để thực hiện crawl cùng lúc nhiều danh sách trang

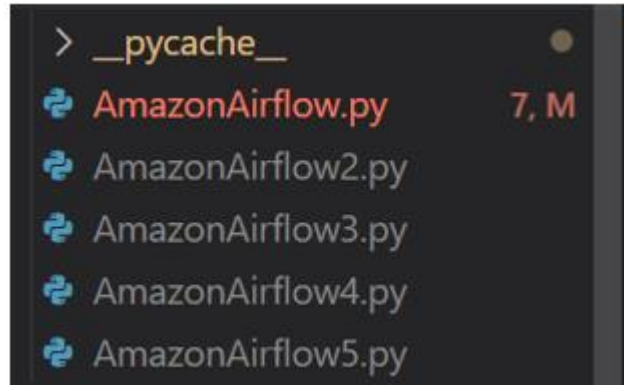


Figure 20 Chia nhiều file thực hiện

- Kết quả File CSV và Json

```
detail_battery_capacity,detail_beschreibung_der_festplatte,detail_besonderes_merkmal,detail_brand,detail_cache_size,d
...ThePhotoStick,,,Gray,"Android devices with OS 6.0 and later, iOS devices with iOS/iPadOS 13.0 and later, Windows
...Gigastone,,,128GB Z40 USB 3.2 OTG Type A+C 2PK,,,,,"Introducing GIGASTONE USB Flash Drives | Elevate Your D
...Easily store and access 2TB of content on the go with the Seagate Portable Drive, a great laptop h
...,"Hardwareverschlüsselung, Rückwärtskompatibel",,,,,,"Reach max performance of PCIe® 4.
...Col 4 detail_brand te al polvo, resistente al agua, portátil, cifrado de hardware",Negro,,,,,Disco SSD,"Your 1
...ThePhotoStick,,,USB,,,,,128 GB,,,,,"Lightweight, Dual USB Connector",,
...SAMSUNG,,,Black,"Laptop, PC, Mac",,"PCIe Gen 5x2, PCIe Gen 4x4",,,,,,"Fast speeds. Power efficiency. Temperat
...,"Get ready to game even faster with the WD_BLACK SN770 NVMe SSD. Zoom past load times and get into
...WD_BLACK,,,Black,Gaming Console,,NVMe,,,,,"Officially licensed PlayStation 5 storage[1], the WD_BLACK SN850P S
...SAMSUNG,,,Titan Gray,"PC, Gaming Console, Tablet, Smartphone, Mac",,USB 3.2 Gen 2,,,,,"""Create. Explore. Stor
...Crucial,,,Black,"Laptop, Desktop",,SATA,,,,,"Ever wonder why your phone responds faster than your computer? It
...iDiskk,,,Silver,"Laptop, Desktop, Camera",,USB,,,,,2 TB,,,,,Mechanical Hard Disk,2.5 Inches,USB 2.0/3.0
...iDiskk,,,silver,"Laptop, Desktop, Camera",,SATA,,,,,2 TB,,,,,Mechanical Hard Disk,2.5 Inches,USB 3.0,,
...fanxiang,,,,"Laptop, Desktop, Tablet, Car Audio, Smartphone",,USB,,,,,256 GB,,,,,Solid State Drive,USB
...UnionSine,,,Black,"PS4, Laptop, Gaming Console, Xbox, Mac",,USB,,,,,1 TB,,,,,Mechanical Hard Disk,2.5 I
...SAMSUNG,,,Black,"Laptop, PC, Mac",,"PCIe Gen 4x4",,,,,,"Unleash the power of the world's No.1 memory flash drive
...WD_BLACK,,,Black,"Laptop, Motherboards",,NVMe,,,,,"Strap in for breakneck gaming speeds with the WD_BLACK SN85
...SAMSUNG,,,Black,"PC, Gaming Console, iPhone 13/16, Tablet, Mac",,USB 3.2 Gen 2,,,,,"Samsung T7 Shield Portable
...SAMSUNG,,,Black,"Laptop, PC, Mac",,SATA 6Gb/s,,,,,"The latest 870 EVO has indisputable performance, reliabilit
```

Figure 21 Kết quả file CSV

```
{
  "title": "ThePhotoStick® Omni 128GB - Secure Photo & Video Backup and Transfer | Digital File Organization | USB
  "price": "$79.99",
  "old_price": "$84.99",
  "product_url": "https://www.amazon.com/sspa/click?ie=UTF8&spc=MT03MjM2NTY8NzIyMzgzODQxOjE3MzQ1MTAxODk6c3BfYXRmX2
  "rating": "",
  "reviews": "6,229",
  "purchases": "10K+ bought in past month",
  "detail_brand": "ThePhotoStick",
  "detail_memory_storage_capacity": "128 GB",
  "detail_hardware_interface": "USB",
  "detail_special_feature": "Lightweight, Dual USB Connector",
  "detail_write_speed": "up to 20 MB/s"
}
```

Figure 22 Kết quả file json

- Merge file CSV sau khi thu thập dữ liệu


```
def merge_csv_files_with_spark(input_directory, output_file='merged_output.csv'):  
    # Khởi tạo SparkSession  
    spark = SparkSession.builder \  
        .appName("CSVFilesMerger") \  
        .getOrCreate()  
  
    # Kiểm tra xem thư mục đầu vào có tồn tại không  
    if not os.path.exists(input_directory):  
        print(f"Error: Directory {input_directory} does not exist.")  
        return 0  
  
    # Danh sách các tệp CSV  
    csv_files = [os.path.join(input_directory, f) for f in os.listdir(input_directory) if f.endswith('.csv')]  
  
    # Kiểm tra nếu không có tệp nào  
    if not csv_files:  
        print("No CSV files found in the directory.")  
        return 0
```

Figure 23 Hàm merge file csv (1)

```
# Đọc và hợp nhất tất cả các tệp CSV  
merged_df = spark.read.option("header", "true").csv(csv_files)  
  
# Tạo thư mục đầu ra nếu chưa tồn tại  
output_directory = os.path.dirname(output_file) or '.'  
os.makedirs(output_directory, exist_ok=True)  
  
# Ghi tệp hợp nhất ra CSV  
merged_df.coalesce(1).write.option("header", "true").csv(output_file, mode="overwrite")  
  
print(f"\nMerging complete!")  
print(f"Total files merged: {len(csv_files)}")  
print(f"Output directory: {output_file}")  
print(f"Total rows in merged file: {merged_df.count()}")  
  
# Dừng SparkSession  
spark.stop()  
  
return len(csv_files)
```

Figure 24 Hàm merge file csv (2)

CHƯƠNG 3: KỸ THUẬT TIỀN XỬ LÝ

3.1. Làm sạch dữ liệu

3.1.1. Loại bỏ cột bị thiếu dữ liệu

- Do dữ liệu bị thiếu nhiều, chỉ có thể giữ lại một số cột đầu có thể khai thác được thông tin từ cột title.

```
columns_to_keep = ["title", "price", "old_price", "rating", "reviews", "purchases", "detail_brand", "detail_memory_storage_capacity", \
                  "detail_hardware_interface", "detail_digital_storage_capacity", "detail_read_speed", "detail_write_speed", "detail_color"]

spark_df = spark_df.select(columns_to_keep)

spark_df.show()
```

Figure 25. Các cột có thể được giữ lại

- Thông qua đánh giá title, các cột trên là các cột có thể khai thác tốt dữ liệu từ cột title. Sau đó ta sẽ đổi tên một số cột được giữ lại để thuận tiện trong quá trình tiền xử lý.

```
spark_df = spark_df.withColumnRenamed("detail_brand", "Brand")\
    .withColumnRenamed("detail_memory_storage_capacity", "Memory Storage Capacity")\
    .withColumnRenamed("detail_hardware_interface", "Hardware Interface")\
    .withColumnRenamed("detail_digital_storage_capacity", "Digital Storage Capacity")\
    .withColumnRenamed("detail_read_speed", "Read Speed")\
    .withColumnRenamed("detail_write_speed", "Write Speed")\
    .withColumnRenamed("detail_color", "Color")

spark_df.show()
```

Figure 26. Đổi tên cột

3.1.2. Xử lý cột price, old_price, rating

- Đối với hai cột price và old_price, ta cần loại bỏ ký hiệu '\$' để có thể chuyển dữ liệu về lại dạng double (hoặc double) để cho thuận tiện công đoạn tính toán trong lúc khai thác dữ liệu.
- Trong lúc xử lý, nếu cột old_price bị thiếu dữ liệu, ta sẽ lấy giá trị từ cột price để điền vào, trong trường hợp price cũng bị thiếu thì ta sẽ bỏ qua và loại hàng dữ liệu đó sau pha làm sạch dữ liệu.
- Đối với cột rating thì ta cần biến kiểu dữ liệu dạng string về lại định dạng

double, nếu dữ liệu không có thì ta sẽ điền giá trị 0.

```
# `price` và `old_price`
# Xóa dấu $ và đổi sang float cho tính toán, trong trường hợp old price không có thì sẽ điền = price
spark_df = spark_df.withColumn("price", F.regexp_replace("price", r'[\$,]', '').cast("float")) \
    .withColumn("old_price", F.when(F.col("old_price").isNull(), F.col("price"))
    .otherwise(F.regexp_replace("old_price", r'[\$,]', '').cast("float")))

# `rating`
# Thay đổi dạng rating
spark_df = spark_df.withColumn("rating",
    F.when(F.col("rating").isNotNull(),
        F.regexp_extract("rating", r"(\d\.\d)", 1).cast("float"))
    .otherwise(F.lit(0.0)))

spark_df.select('title', 'price', 'old_price', 'rating').show()
```

Figure 27. Xử lý cột price, old_price và rating

title	price	old_price	rating
NULL	NULL	NULL	0.0
MFi Certified 256...	39.99	39.99	0.0
fanxiang 256GB Fl...	36.99	39.99	0.0
ThePhotoStick® Om...	92.96	149.99	0.0
Seagate Portable ...	69.99	79.99	4.599999904632568
SAMSUNG 990 PRO S...	169.86	249.99	0.0
Seagate Storage E...	199.99	359.99	4.599999904632568
NULL	NULL	NULL	0.0
SAMSUNG 980 PRO S...	169.0	198.75	0.0
SAMSUNG 990 EVO S...	73.0	149.99	4.699999809265137
SanDisk 4TB Extre...	259.99	359.99	4.599999904632568
WD_BLACK 2TB SN85...	149.99	189.99	0.0
SAMSUNG T7 Shield...	271.86	499.99	4.699999809265137
Western Digital W...	119.99	159.99	0.0
NULL	NULL	NULL	0.0
JSL JDTCDC 256GB 3...	89.99	109.99	4.300000190734863
1TB External SSD,...	79.99	79.99	4.599999904632568
MOSDART 64GB USB ...	9.99	13.99	4.599999904632568
WD_BLACK 2TB SN85...	179.99	229.99	0.0
SAMSUNG T7 Portab...	278.14	278.14	0.0

only showing top 20 rows

Figure 28. Kết quả price, old_price và rating

3.1.3. Xử lý cột reviews, purchases

- Đối với cột reviews, ta chỉ cần đổi lại dữ liệu thành dạng số (int).
- Đối với cột purchases, kiểu dữ liệu ban đầu là string, số lượng sản phẩm bán ra được ở tháng trước (VD: 10K+ bought in last month). Cần phải lấy giá trị ra và chuyển đổi về dạng số.

```
# `reviews`  
# Đổi số lượng review thành số  
spark_df = spark_df.withColumn("reviews",  
                                F.when(F.col("reviews").isNotNull(),  
                                       F.regexp_replace(F.col("reviews"), ",", "").cast("int"))  
                                .otherwise(F.lit(0)))  
  
# `purchases`  
# Lấy thông tin cần thiết từ chuỗi và biến đổi thành integer  
spark_df = spark_df.withColumn("purchases",  
                                F.when(F.col("purchases").rlike(r"(\d+K?)\+ bought in past month"),  
                                       F.regexp_replace(F.regexp_extract(F.col("purchases"), r"(\d+K?)", 1), "K", "000")  
                                       .cast("int"))  
                                .otherwise(F.lit(0)))  
  
spark_df.select('reviews', 'purchases').show()
```

Figure 29. Xử lý cột reviews, purchases

```

+-----+-----+
|reviews|purchases|
+-----+-----+
|      0|        0|
|     96|     500|
|    204|        0|
|   2182|    3000|
| 261799|   10000|
|   10202|   10000|
|   23493|   10000|
|        0|        0|
|   39332|   10000|
|    1591|   10000|
|   70295|    9000|
|   22658|   10000|
|   13718|    7000|
|   19165|    9000|
|        0|        0|
|     58|        0|
|    160|     50|
|   13954|    400|
|    5423|    7000|
|   33496|    5000|
+-----+-----+
only showing top 20 rows

```

Figure 30. Kết quả reviews, purchases

3.1.4. Xử lý cột Brands

Brand	count
HUPINS	1
Thales	1
OneKey	1
Accessonico	2
Garrulax	14
Flylin	2
Sanakma	1
Rpanle	1
Iomega	4
QNAP	22
Omni DataSafe	1
Hiseeu	3
VAVA	1
ELUTENG	2
Apricorn	15
KINGDATA	1
15	1
V7	1
Acxico	1
Patriot Memory	16
THKAILAR	5
CaSZLUTION	3
...	
KIWIFOTOS	1
CASIMY	1

Figure 31. Đếm số Brands

- Dựa trên hình, ta thấy cột Brand nhiều hãng lớn nhỏ khác nhau, điều này có thể dẫn tới nhiều dữ liệu do quá nhiều giá trị nên trước tiên, ta sẽ lọc bớt các hãng nhỏ lẻ chỉ xuất hiện vài lần với số lần xuất hiện phải ít nhất trên 10 lần mới được chấp nhận (Threshold > 10).

```

Tabnine | Edit | Test | Explain | Document | Ask
def filter_brands_by_count(df, threshold):
    brand_counts = df.groupBy("Brand").count()
    filtered_brands = brand_counts.filter(F.col("count") > threshold).select("Brand")
    return [row["Brand"] for row in filtered_brands.collect()]

threshold = 10

accepted_brands = filter_brands_by_count(spark_df, threshold)

accepted_brands

```

Figure 32. Lọc Brands

- Mảng `accepted_brands` chứa các hãng xuất hiện trên 10 lần, từ đây, nếu giá trị là trống thì trước tiên sẽ kiểm dữ liệu trong title và so sánh với mảng `accepted_brands`, nếu phù hợp sẽ điền vào giá trị còn nếu không thì sẽ điền 'others' đại diện cho các hãng nhỏ hoặc không có. Trong trường hợp dữ liệu đã có thì sẽ so sánh với mảng, nếu không hợp sẽ tương tự điền vào 'others'.
- Sau khi hoàn thành bước trên thì ta sẽ chuẩn hóa cột theo lowercase.

```

# Xử lý Brand
spark_df = spark_df.withColumn(
    "Brand",
    F.coalesce(
        *[
            F.when(F.col("title").contains(brand), brand)
            for brand in accepted_brands
        ],
        F.lit("others"),
        F.col("Brand")
    )
)

#Chuẩn hóa lowercase
spark_df = spark_df.withColumn("Brand", lower(spark_df["Brand"]))
spark_df.select('Brand').show(spark_df.count())

```

Figure 33. Xử lý cột Brands

Brand
others
others
fanxiang
thephotostick
seagate
samsung
seagate
others
samsung
samsung
sandisk
wd_black
samsung
wd_black
others
jsl jdtcd
others
others
wd_black
samsung
crucial
thephotostick
...
others
others

Figure 34. Kết quả Brands

3.1.5. Xử lý cột Memory Storage Capacity, Digital Storage Capacity

- Do dữ liệu của sản phẩm bộ nhớ được chia thành hai loại là USB và SSD, HDD, nên tương tự ta sẽ có Memory Storage Capacity chứa thông tin về độ lớn của USB và Digital Storage Capacity.
- Tương tự như chiến thuật xử lý đề ra ban đầu, ta sẽ tìm dữ liệu trong title trong trường hợp bị thiếu, nhưng trước tiên cần so sánh xem sản phẩm đó thuộc loại nào.
- Sau khi toàn bộ dữ liệu được điền vào, ta sẽ chuẩn hóa toàn bộ về đơn vị GB. Ngoài ra, nếu sản phẩm là có dữ liệu một bên và không có bên còn lại,

giá trị 0 sẽ được điền vào chỗ trống.

```
# Xác định Pattern cho USB và SSD, HDD
usb_pattern = r"(?i)\bUSB\b"
ssd_patterns = r"(?i)\bSSD\b|\bSolid State Drive\b|\bPortable External Hard Drive\b|\bHDD\b"

# Lấy chỉ số dữ liệu Storage Capacity từ title nếu dữ liệu bị thiếu
# Memory Storage Capacity cho USB và Digital Storage Capacity cho SSD, HDD Portable External Hard Drive
spark_df = (
    spark_df
    .withColumn(
        "Memory Storage Capacity",
        F.when(
            F.col("Memory Storage Capacity").isNull() & F.col("title").rlike(usb_pattern),
            F.regexp_extract(F.col("title"), r"(\d+(?:\.\d+)?)\s?(TB|GB|MB)", 0)
        ).otherwise(F.col("Memory Storage Capacity"))
    )
    .withColumn(
        "Digital Storage Capacity",
        F.when(
            F.col("Digital Storage Capacity").isNull() & F.col("title").rlike(ssd_patterns),
            F.regexp_extract(F.col("title"), r"(\d+(?:\.\d+)?)\s?(TB|GB|MB)", 0)
        ).otherwise(F.col("Digital Storage Capacity"))
    )
)

spark_df.select('Memory Storage Capacity', 'Digital Storage Capacity').show()
```

Figure 35. Xử lý cột Memory Storage Capacity, Digital Storage Capacity (1)

```
# Xóa các khoảng trống dư thừa giữa giá trị và đơn vị
spark_df = spark_df.withColumn(
    "Memory Storage Capacity",
    F.regexp_replace(F.col("Memory Storage Capacity"), "\\s+", "")
).withColumn(
    "Digital Storage Capacity",
    F.regexp_replace(F.col("Digital Storage Capacity"), "\\s+", "")
)

# Đẩy giá trị 0 vào các cột trống
spark_df = spark_df.fillna("0", subset=["Memory Storage Capacity", "Digital Storage Capacity"])
```

Figure 36. Xử lý cột Memory Storage Capacity, Digital Storage Capacity (2)

```

unit_multiplier = {
    "TB": 1024,
    "GB": 1,
    "MB": 1/1024
}

# Thêm cột mới cho phần giá trị
spark_df = spark_df \
    .withColumn("numeric_value", regexp_extract(col("Digital Storage Capacity"), r"(\d+\.\d*)", 1)
    .cast("float"))

# Thêm cột mới cho phần
spark_df = spark_df \
    .withColumn("unit", regexp_extract(col("Digital Storage Capacity"), r"([A-Za-z]+)", 1))

# Chuẩn hóa sang dạng GB
spark_df = spark_df.withColumn(
    "Digital Storage Capacity (GB)",
    when(col("numeric_value") == 0, 0)
    .when(col("unit") == "TB", col("numeric_value") * unit_multiplier["TB"])
    .when(col("unit") == "GB", col("numeric_value") * unit_multiplier["GB"])
    .when(col("unit") == "MB", col("numeric_value") * unit_multiplier["MB"])
    .otherwise(None)
)

spark_df.select('Digital Storage Capacity', 'Digital Storage Capacity (GB)').show()

```

Figure 37. Xử lý cột Memory Storage Capacity, Digital Storage Capacity (3)

```

# Tương tự cho Memory Storage Capacity
spark_df = spark_df.withColumn("numeric_value_mem", regexp_extract(col("Memory Storage Capacity"), r"(\d+\.\d*)", 1).cast("float"))
spark_df = spark_df.withColumn("unit_mem", regexp_extract(col("Memory Storage Capacity"), r"([A-Za-z]+)", 1))
spark_df = spark_df.withColumn(
    "Memory Storage Capacity (GB)",
    when(col("numeric_value_mem") == 0, 0) # Keep zero values unchanged
    .when(col("unit_mem") == "TB", col("numeric_value_mem") * unit_multiplier["TB"])
    .when(col("unit_mem") == "GB", col("numeric_value_mem") * unit_multiplier["GB"])
    .when(col("unit_mem") == "MB", col("numeric_value_mem") * unit_multiplier["MB"])
    .otherwise(None)
)

# Bỏ các cột không cần thiết
spark_df = spark_df.drop("numeric_value", "unit", "numeric_value_mem", "unit_mem", "Digital Storage Capacity", "Memory Storage Capacity")

```

Figure 38. Xử lý cột Memory Storage Capacity, Digital Storage Capacity (4)

Memory Storage Capacity (GB)	Digital Storage Capacity (GB)
0.0	0.0
256.0	0.0
256.0	256.0
256.0	0.0
2048.0	2048.0
0.0	2048.0
0.0	2000.0
0.0	0.0
0.0	2048.0
0.0	1024.0
4096.0	4096.0
0.0	2000.0
0.0	4096.0
0.0	2000.0
0.0	0.0
256.0	0.0
1024.0	1024.0
64.0	0.0
0.0	2048.0
4096.0	4096.0

only showing top 20 rows

Figure 39. Kết quả Memory Storage Capacity, Digital Storage Capacity

3.1.6. Xử lý cột Hardware Interface

- Tương tự cột Brand nhưng không cần phải lọc ra các Interface do số lượng Interface nhỏ.

```
# 'Hardware Interface pattern'
hardware_interfaces = ["USB 3.0", "USB", "USB 3.2 Gen 1", "USB 2.0", "USB 3.2 Gen 2", "3.2 USB"]

# Xóa ',' và tạo cột mới temp_title để sử dụng cho extract hardware_interfaces
spark_df = spark_df.withColumn("temp_title", F.regexp_replace(F.col("title"), "[,]", ""))

hardware_interface_expr = F.coalesce(
    *[F.when(F.col("temp_title").contains(interface), interface) for interface in hardware_interfaces],
    F.lit("USB")
)

# Hardware Interface
spark_df = spark_df.withColumn(
    "Hardware Interface",
    F.when(F.col("Hardware Interface").isNull(), hardware_interface_expr)
    .otherwise(F.col("Hardware Interface"))
)

# Re-check cho các trường hợp sai dữ liệu, set default là USB nếu không phù hợp pattern
spark_df = spark_df.withColumn(
    "Hardware Interface",
    F.when(
        # If Hardware Interface is still null or doesn't match any known hardware interface
        (F.col("Hardware Interface").isNull()) |
        (~F.col("Hardware Interface").isin(hardware_interfaces)),
        # Then re-run extraction or set default
        F.coalesce(
            *[F.when(F.col("temp_title").contains(interface), interface) for interface in hardware_interfaces],
            F.lit("USB")
        )
    ).otherwise(F.col("Hardware Interface"))
)
```

Figure 40. Xử lý cột Hardware Interface

```

+-----+-----+
|          temp_title|Hardware Interface|
+-----+-----+
|          NULL|          USB|
|MFi Certified 256...|          USB|
|fanxiang 256GB Fl...|          USB|
|ThePhotoStick® Om...|          USB|
|Seagate Portable ...|        USB 3.0|
|SAMSUNG 990 PRO S...|          USB|
|Seagate Storage E...|          USB|
|          NULL|          USB|
|SAMSUNG 980 PRO S...|          USB|
|SAMSUNG 990 EVO S...|          USB|
|SanDisk 4TB Extre...|          USB|
|WD_BLACK 2TB SN85...|          USB|
|SAMSUNG T7 Shield...|          USB|
|Western Digital W...|          USB|
|          NULL|          USB|
|JSL JDTC 256GB 3...|          USB|
|1TB External SSD ...|          USB|
|MOSDART 64GB USB ...|        USB 2.0|
|WD_BLACK 2TB SN85...|          USB|
|SAMSUNG T7 Portab...|          USB|
|Crucial P3 Plus 4...|          USB|
|ThePhotoStick® Om...|          USB|
|...
|SanDisk 256GB Ult...|        USB 3.0|
+-----+-----+
only showing top 67 rows

```

Figure 41. Kết quả Hardware Interface

3.1.7. Xử lý cột Write Speed, Read Speed

- Tương tự như cột Memory Storage Capacity, ta sẽ tìm dữ liệu trong title nếu dữ liệu trống và bỏ qua nếu dữ liệu đã có. Trong trường hợp nếu title không có dữ liệu thì sẽ điền 0.
- Đối với Read Speed, tương tự như Write Speed nhưng trong trường hợp dữ liệu bị thiếu thì ta sẽ lấy dữ liệu của Write Speed điền vào Read Speed.

```
# Pattern cho Write Speed
write_speed_pattern = r"(\d{1,3}(?:,\d{3})*(?:\.\d+)?)(?=\s?(MB/s|Megabytes Per Second|MB per second))"

# Xử lý Write Speed
spark_df = spark_df.withColumn(
    "Write Speed",
    F.when(
        F.col("Write Speed").rlike(write_speed_pattern),
        F.regexp_replace(F.regexp_extract(F.col("Write Speed"), write_speed_pattern, 1), ",", "").cast("float")
    ).when(
        F.col("title").rlike(write_speed_pattern),
        F.regexp_replace(F.regexp_extract(F.col("title"), write_speed_pattern, 1), ",", "").cast("float")
    ).otherwise(F.lit(0))
)

spark_df.select('title', 'Write Speed').show(truncate = True)
```

Figure 42. Xử lý cột Write Speed

title	Write Speed
NULL	0.0
MFi Certified 256GB	0.0
fanxiang 256GB F100	490.0
ThePhotoStick® Omnia	0.0
Seagate Portable 8TB	0.0
SAMSUNG 990 PRO SSD	7450.0
Seagate Storage Expansion	0.0
NULL	0.0
SAMSUNG 980 PRO SSD	0.0
SAMSUNG 990 EVO SSD	5000.0
SanDisk 4TB Extreme	50.0
WD_BLACK 2TB SN850	7300.0
SAMSUNG T7 Shield	50.0
Western Digital WD	5150.0
NULL	0.0
JSL JD TDC 256GB 3.5"	0.0
1TB External SSD, 100MB/s	0.0
MOSDART 64GB USB 3.1	0.0
WD_BLACK 2TB SN850	7300.0
SAMSUNG T7 Portable	1050.0

only showing top 20 rows

Figure 43. Kết quả Write Speed

```
# Xử lý Read Speed
read_speed_pattern = r"(\d{1,3}(?:,\d{3})*(?:\.\d+)?)(?=\s?(MB/s|Megabytes Per Second|MB per second))"

spark_df = spark_df.withColumn(
    "Read Speed",
    F.when(
        F.col("Read Speed").rlike(read_speed_pattern),
        F.regexp_replace(F.regexp_extract(F.col("Read Speed"), read_speed_pattern, 1), ",", "").cast("float")
    ).otherwise(F.col("Write Speed")) # Sử dụng Write Speed trong trường hợp không có giá trị
)
```

Figure 44. Xử lý cột Read Speed

title	Read Speed
NULL	0.0
MFi Certified 256GB	0.0
fanxiang 256GB Fl...	560.0
ThePhotoStick® Om...	350.0
Seagate Portable ...	0.0
SAMSUNG 990 PRO S...	7450.0
Seagate Storage E...	0.0
NULL	0.0
SAMSUNG 980 PRO S...	0.0
SAMSUNG 990 EVO S...	5000.0
SanDisk 4TB Extre...	50.0
WD_BLACK 2TB SN85...	7300.0
SAMSUNG T7 Shield...	50.0
Western Digital W...	5150.0
NULL	0.0
JSL JDTC 256GB 3...	0.0
1TB External SSD,...	0.0
MOSDART 64GB USB ...	0.0
WD_BLACK 2TB SN85...	7300.0
SAMSUNG T7 Portab...	1050.0

only showing top 20 rows

Figure 45. Kết quả Read Speed

3.1.8. Xử lý cột Color

- Tương tự như Brands, đánh giá sơ bộ cho ta thấy không những xuất hiện lượng lớn Color khác nhau, cột cũng bị nhiễu dữ liệu do các dữ liệu không

liên quan xuất hiện.

Color	count
NAS SSD 4TB	1
Black / Red	1
512gb pink	3
RED	1
Dark Grey	3
PCIe Gen4 NVMe SS...	1
disk; disc; stora...	1
"2.5"/7mm SSD"	1
Aluminium	1
Gaming Console, C...	12
Dark Grayish Brown	1
Silvery white	1
Black NVME 2230 t...	1
Z2S	1
orange	3
NAS SSD 2TB 2-Pack	1
2.5 Inches	4
BLACK	6
Old Generation Su...	2
PCIe Gen4 NVMe SS...	1
ZR RGB Lighting	5
NV7000 Metal ra...	1
...	
Multiple	1
1T	2

Figure 46. Đếm số Color

- Bắt buộc phải tạo hẵn một mảng chứa các màu định sẵn để xử lý.


```

color_pattern = r"(?i)(Black|White|Gray|Red|Blue|Green|Yellow|Pink|Silver|Gold)" # Thêm nếu cần

# Xử lý cho trường hợp đã có dữ liệu
# Thêm nếu cần
accepted_colors = ["Black", "White", "Gray", "Red", "Blue", "Green", "Yellow", "Pink",
                   "Silver", "Gold", "Space Gray", "Titan Gray", "Champagne Gold", "Brown", "multi-color"]

# `Color`
spark_df = spark_df.withColumn(
    "Color",
    F.when(
        F.col("Color").isNull() & F.col("title").rlike(color_pattern), # Lấy dữ liệu từ Title nếu Color NULL
        F.lower(F.regexp_extract(F.col("title"), color_pattern, 1)) # Chuẩn hóa lowercase
    ).when(
        F.col("Color").isNull(), # Trong trường hợp Title không có, thế = "others"
        F.lit("others")
    ).when(
        ~F.col("Color").isin(accepted_colors), # Nếu dữ liệu có sẵn không phù hợp pattern thì thế = "others"
        F.lit("others")
    ).otherwise(
        F.lower(F.col("Color")) # Nếu hợp thì chuẩn hóa
    )
)

```

Figure 47. Xử lý cột Color

Color	count
green	46
yellow	1
champagne gold	1
silver	220
white	60
gray	15
others	2166
pink	80
red	111
gold	25
multi-color	5
titan gray	2
black	829
brown	1
space gray	9
blue	169

Figure 48. Kết quả Color

3.1.9. Loại bỏ các hàng bị thiếu dữ liệu và bị lặp

```
spark_df = spark_df.dropDuplicates()
spark_df = spark_df.na.drop()
spark_df.show()
```

Figure 49. Loại bỏ hàng bị lặp và bị thiếu dữ liệu

	title	price	old_price	rating	reviews	purchases	Brand	hardware	Interface	Read Speed	Write Speed	Color	Digital Storage Capacity (GB)	Memory Storage Capacity (GB)
1	7SL 3D1DC 256GB 3...	89.99	109.99	4.300000	190734863	58	jsl	jdktr	USB	0.0	0.0	others	0.0	256.0
2	USB C Flash Drive...	27.99	29.39	4.300000	190734863	47	woolken	USB 2.0	18.0	18.0	0.0	others	0.0	16.0
3	Avolution PRO-SU ...	78.99	78.99	0.0	0	0	avolution	USB 3.0	0.0	0.0	0.0	white	3072.0	3072.0
4	SCINCE 512GB Fla...	52.99	52.99	4.699999	989265137	16	scince	USB	30.0	30.0	0.0	red	0.0	512.0
5	Apricorn 20TB Aeg...	1055.29	1155.42	0.0	0	0	apricorn	USB 3.0	0.0	0.0	0.0	black	20000.0	20480.0
6	Gigastone SATA S...	59.99	59.99	4.400000	95367432	120	gigastone	USB	530.0	530.0	0.0	others	1000.0	0.0
7	FN908 2TB M.2 228...	122.99	122.99	4.599999	984632568	1024	others	USB	800.0	800.0	0.0	green	2048.0	0.0
8	Transcend 1TB MTS...	99.6	99.6	4.699999	989265137	107	transcend	USB	500.0	500.0	0.0	others	1024.0	0.0
9	HD Purple 6TB Sur...	138.59	138.59	4.400000	95367432	172	wd	USB	0.0	0.0	0.0	others	6144.0	0.0
10	Avolution H02500D...	24.5	24.5	4.400000	95367432	438	avolution	USB 3.0	0.0	0.0	0.0	black	500.0	500.0
11	64 GB Flash Drive...	29.99	29.99	0.0	0	0	others	USB 2.0	0.0	0.0	0.0	black	0.0	64.0
12	Mansenda 128GB US...	17.88	17.88	0.0	525	0	others	USB 3.0	40.0	40.0	0.0	blue	0.0	128.0
13	Seagate Cheetah 1...	180.84	180.84	0.0	169	0	seagate	USB	0.0	0.0	0.0	green	600.0	0.0
14	Western Digital ...	49.99	73.84	0.0	84	0	western digital	USB	0.0	0.0	0.0	others	500.0	0.0
15	500GB Portable Ex...	49.99	49.99	0.0	667	0	others	USB	430.0	430.0	0.0	black	500.0	500.0
16	Avolution PRO-SX ...	76.99	76.99	4.699999	989265137	76	avolution	USB 3.0	0.0	0.0	0.0	others	3000.0	3072.0
17	Samsung T7 Portab...	249.23	249.23	0.0	1319	0	others	USB	0.0	0.0	0.0	red	2048.0	2048.0
18	Patriot Burst El...	124.0	139.0	4.599999	984632568	275	others	USB	0.0	0.0	0.0	others	240.0	0.0
19	Phoxfer RecovStic...	19.99	19.99	0.0	85	100	others	USB	40.0	40.0	0.0	others	0.0	64.0
20	USB 16GB Flash Dr...	29.88	29.88	4.300000	190734863	1082	eastbull	USB	0.0	0.0	0.0	others	0.0	16.0

Figure 50. Kết quả dataframe sau làm sạch dữ liệu

3.2. Chuẩn bị dữ liệu cho kỹ thuật khai thác

3.2.1. Kỹ thuật MinMaxScaler:

- Min/Max scaling là kỹ thuật lấy từng giá trị trừ đi cho giá trị tối thiểu sau đó chia cho hiệu giá trị lớn nhất và nhỏ nhất.

$$X_{new} = \frac{X_i - X_{min}}{X_{max} - X_{min}}$$

- Kỹ thuật này chia tỷ lệ lại một đặc tính hoặc giá trị quan sát với giá trị phân phối từ 0 đến 1.

```

# Các feature cần sử dụng MinMaxScaler
numerical_features = ["price", "old_price", "rating", "reviews",
                      "purchases", "Write Speed", "Read Speed",
                      "Digital Storage Capacity (GB)", "Memory Storage Capacity (GB)"]

for feature in numerical_features:
    window_spec = Window.partitionBy()

    # Tính toán min, max thông qua Window
    spark_df = spark_df.withColumn("feature_min", min(col(feature)).over(window_spec))
    spark_df = spark_df.withColumn("feature_max", max(col(feature)).over(window_spec))

    # Áp dụng MinMaxScaler lên cột features
    spark_df = spark_df.withColumn(
        feature,
        F.when((col("feature_max") - col("feature_min")) != 0,
               (col(feature) - col("feature_min")) / (col("feature_max") - col("feature_min")))
               .otherwise(lit(0))
    )

    # Drop các cột tạm
    spark_df = spark_df.drop("feature_min", "feature_max")

```

Figure 51. Kỹ thuật MinMaxScaler lên các cột dữ liệu số

3.2.2. Kỹ thuật StringIndex + One-hot Encoding:

- String Indexer là một kỹ thuật được sử dụng để chuyển đổi các giá trị chuỗi (categorical data) thành giá trị số (numeric data) bằng cách ánh xạ từng giá trị chuỗi trong cột thành một giá trị số nguyên. Ví dụ: ["apple", "banana", "orange"] -> ["0", "1", "2"]. Quá trình này sắp xếp các chuỗi dựa trên tần suất xuất hiện của chúng (giá trị có tần suất cao nhất sẽ được ánh xạ vào số nhỏ nhất, thường là 0).
- One-hot encoding là kỹ thuật chuyển đổi dữ liệu phân loại thành dạng vector nhị phân. Với một cột có n giá trị khác nhau (categories), One-hot encoding tạo ra n cột mới, trong đó mỗi cột biểu diễn một giá trị duy nhất và chứa giá trị 0 hoặc 1.

$$\begin{matrix} apple \\ banana \\ orange \end{matrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

```
Tabnine | Edit | Test | Explain | Document | Ask
def one_hot_encode(df, column_name):
    # Do OneHotEncoder của PySpark chỉ hoạt động trên dữ liệu số nên cần hỗ trợ của StringIndexer để đổi
    string_indexer = StringIndexer(inputCol=column_name, outputCol=f"{column_name} Index")

    # Áp dụng OneHotEncoder cho mỗi danh mục trong cột
    one_hot_encoder = OneHotEncoder(inputCol=f"{column_name} Index", outputCol=f"{column_name} Encoded")

    # Tạo và chạy pipeline
    pipeline = Pipeline(stages=[string_indexer, one_hot_encoder])
    model = pipeline.fit(df)

    # Biến đổi dataframe
    df_transformed = model.transform(df)

    # Xóa cột Index trung gian (optional)
    # df_transformed = df_transformed.drop(f"{column_name} Index")

    return df_transformed

spark_df = one_hot_encode(spark_df, "Color")
spark_df = one_hot_encode(spark_df, "Hardware Interface")
spark_df = one_hot_encode(spark_df, "Brand")
```

Figure 52. StringIndexer + One-hot Encoding

CHƯƠNG 4: KỸ THUẬT KHAI THÁC DỮ LIỆU

4.1. K-Means

4.1.1. Cơ sở lý thuyết

- Thuật ngữ "K-means" được James MacQueen sử dụng lần đầu tiên vào năm 1967, mặc dù ý tưởng này quay trở lại Hugo Steinhaus vào năm 1956. Thuật toán tiêu chuẩn được đề xuất lần đầu tiên bởi Stuart Lloyd của Bell Labs vào năm 1957 như một kỹ thuật cho điều chế mã xung, mặc dù nó không được xuất bản dưới dạng một bài báo cho đến năm 1982. Năm 1965, Edward W. Forgy đã công bố về cơ bản cùng một phương pháp, đó là lý do tại sao nó đôi khi được gọi là Lloyd-Forgy.
- K-Means gán các điểm dữ liệu vào một trong K cụm (clusters) dựa trên khoảng cách của chúng đến tâm cụm (**centroid**). Thuật toán bắt đầu bằng cách chọn ngẫu nhiên các tâm cụm trong không gian dữ liệu. Sau đó, mỗi điểm dữ liệu được gán vào một cụm dựa trên khoảng cách của nó tới tâm cụm gần nhất. Sau khi tất cả điểm dữ liệu được gán cụm, tâm cụm mới được tính lại. Quá trình này lặp đi lặp lại cho đến khi tìm được các cụm tối ưu.
- Trong phân tích này, giả định rằng số lượng cụm K đã được xác định trước và chúng ta cần phân loại các điểm dữ liệu vào một trong các nhóm đó.
- Trong một số trường hợp, **K** không được xác định rõ ràng, và chúng ta cần xác định số lượng cụm K tối ưu. K-means hoạt động tốt nhất khi dữ liệu được phân tách rõ ràng. Khi các điểm dữ liệu chồng lấn nhau, K-means không phải là phương pháp phù hợp. Thuật toán này nhanh hơn so với các kỹ thuật gom cụm khác và tạo sự liên kết chặt chẽ giữa các điểm dữ liệu.
- Tuy nhiên, K-means có một số hạn chế:
 - Không cung cấp thông tin rõ ràng về chất lượng cụm.
 - Kết quả cụm phụ thuộc vào cách khởi tạo tâm cụm ban đầu, có thể dẫn đến các kết quả khác nhau.

- Thuật toán nhạy cảm với nhiễu (noise) và có thể bị kẹt ở cực tiểu cục bộ (local minima).
- Mục tiêu của thuật toán K-Means: Mục tiêu của việc gom cụm (**clustering**) là chia tập dữ liệu hoặc quần thể thành các nhóm sao cho các điểm dữ liệu trong cùng một nhóm có tính tương đồng cao hơn so với các điểm dữ liệu trong các nhóm khác. Về cơ bản, đây là một quá trình phân nhóm dựa trên độ tương tự và khác biệt giữa các điểm dữ liệu.
- K-Means hoạt động như thế nào? Giả sử chúng ta có một tập dữ liệu gồm các đối tượng, mỗi đối tượng có các đặc trưng (features) với các giá trị tương ứng (dạng vector). Nhiệm vụ của chúng ta là phân loại các đối tượng này vào các nhóm cụ thể. Để làm điều này, chúng ta sử dụng thuật toán **K-means**, một thuật toán học máy không giám sát. Chữ **K** trong tên thuật toán đại diện cho số lượng nhóm/cụm mà chúng ta muốn phân loại các đối tượng vào
- Giải thuật K-means được thực hiện như sau

- Khởi tạo các trung tâm cụm ban đầu

$$\mathcal{C}^{(0)} = \{ m_1^{(0)}, m_2^{(0)}, \dots, m_k^{(0)} \}$$


- k là số lượng cụm được xác định trước.

- Phân cụm dữ liệu

- Với mỗi điểm dữ liệu, ta sẽ tính khoảng cách của nó tới các trung tâm (bằng Khoảng cách Euclid). Ta sẽ gán chúng vào trung tâm gần nhất. Tập hợp các điểm được gán vào cùng 1 trung tâm sẽ tạo thành cụm.

$$d(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

- Trong đó

 d(x,y): Khoảng cách giữa hai điểm x và y.

✚ n: Số chiều (số thành phần) của không gian.

✚ x_i và y_i : Thành phần thứ i của điểm x và y .

- Cập nhật trung tâm cụm
 - Sau khi gán các điểm dữ liệu vào các cụm, trung tâm cụm $m(i)$ mới sẽ được tính lại bằng trung bình cộng của tất cả các điểm dữ liệu thuộc cụm i :

$$m_i^{(t+1)} = \frac{1}{|C_i|} \sum_{x_j \in C_i} x_j$$

- Trong đó

✚ $(m_i^{(t+1)})$: Trung tâm cụm mới ở lần lặp thứ $(t + 1)$.

✚ (C_i) : Tập hợp các điểm dữ liệu thuộc cụm (i) .

✚ $(|C_i|)$: Số lượng điểm trong cụm (i) .

```
# Stop if centroids have minimal movement
if max_centroid_shift < 0.0001:
    break
```

Figure 53 Điều kiện dừng

- Kiểm tra điều kiện dừng: Thuật toán dừng khi các trung tâm cụm ổn định, tức là sự thay đổi giữa các trung tâm nhỏ hơn một ngưỡng nhất định.

4.1.2. Thực nghiệm

```
def calculate_distance(point1, point2):
    return sum((x - y) ** 2 for x, y in zip(point1, point2))
```

- Hàm `calculate_distance(point1, point2)` dùng để tính khoảng cách Euclidean

giữa hai điểm dữ liệu point1 và point2:

- Dùng công thức $\sqrt{\sum (x_i - y_i)^2}$
- Trong code, công thức này được tính bằng cách lặp qua các cặp tọa độ tương ứng (x, y) của hai điểm và tính tổng bình phương hiệu của chúng.

```
class K_Mean:
    def __init__(self, k=2, predictCol='prediction', seed=1):
        self.k = k
        self.centroids = None
        self.predictCol = predictCol
        self.seed = seed
```

- Khởi tạo class K-Means
- `__init__`: Hàm khởi tạo với các tham số:
 - k: số cụm cần phân
 - predictCol: Tên cột sẽ chứa giá trị dự đoán cụm (mặc định là 'prediction')
 - seed: Dùng để đảm bảo tính ngẫu nhiên có thể tái lập (random seed)
 - centroids: Lưu trữ danh sách tọa độ của các tâm cụm (**centroid**)

```
def find_closest_cluster(self, data_point):
    distances = [calculate_distance(data_point, centroid) for centroid
in self.centroids]
    return distances.index(min(distances))
```

- Hàm `find_closest_cluster` nhằm xác định cụm gần nhất cho một điểm dữ liệu (data_point)

- Cách thực hiện:
 - o Tính khoảng cách từ điểm dữ liệu đến từng tâm cụm.
 - o Trả về chỉ số của cụm có khoảng cách nhỏ nhất.

```
def fit(self, dataframe, feature_columns):  
    df = dataframe.withColumn('features', F.array(feature_columns))
```

- Hàm này thực hiện quá trình huấn luyện (training) mô hình K-Means Clustering trên dữ liệu. Dưới đây là giải thích chi tiết từng dòng:
- dataframe: Tập dữ liệu đầu vào dưới dạng DataFrame của PySpark.
- feature_columns: Danh sách các cột đặc trưng cần sử dụng để gom cụm.
- withColumn('features', F.array(...)): Tạo cột mới tên là features, chứa mảng (array) các giá trị của các cột đặc trưng trong feature_columns.

```
for _ in range(20):  
    # Randomly select initial centroids  
    self.centroids = df.select('features').rdd.map(lambda r:  
r.features).takeSample(False, num=self.k, seed=self.seed)  
    self.seed += 1
```

- Lặp 20 lần để đảm bảo việc chọn tâm cụm ban đầu có chất lượng tốt hơn:
- df.select('features'): Lấy cột features từ DataFrame.
- rdd.map(lambda r: r.features): Chuyển đổi DataFrame thành RDD, chỉ lấy các giá trị của cột features.
- takeSample(False, num=self.k, seed=self.seed): Lấy ngẫu nhiên k điểm từ tập dữ liệu làm tâm cụm ban đầu, sử dụng seed để tạo tính ngẫu nhiên tái lập.

- `self.seed += 1`: Tăng giá trị seed sau mỗi lần khởi tạo để tránh việc lặp lại cùng một mẫu tâm cụm.

```
while True:
```

```
    # Create a user-defined function to assign clusters
```

```
    assign_cluster_udf = F.udf(self.find_closest_cluster,  
IntegerType())
```

- Vòng lặp chính: Chạy đến khi các tâm cụm hội tụ (không còn thay đổi đáng kể).
- `assign_cluster_udf`: Tạo một UDF (hàm do người dùng định nghĩa), sử dụng hàm `find_closest_cluster` để gán mỗi điểm dữ liệu vào cụm gần nhất.

```
    # Assign each point to the nearest cluster
```

```
    df = df.withColumn( self.predictCol, assign_cluster_udf(F.col('features')))
```

- `withColumn(self.predictCol, ...)`: Thêm một cột mới (`self.predictCol`, mặc định là `prediction`) vào `DataFrame`.
- `assign_cluster_udf(F.col('features'))`: Tính toán cụm gần nhất cho mỗi điểm dựa trên vector features.

```
# Recalculate cluster centroids
```

```
cluster_averages = (  
    df.rdd  
    .map(lambda row: (row[self.predictCol], row['features']))  
    .groupByKey()  
    .mapValues(lambda features: [  
        sum(dimension) / len(features)  
        for dimension in zip(*features))
```

```
    )
)
```

- df.rdd: Chuyển đổi DataFrame thành RDD để thao tác theo kiểu phân tán, hỗ trợ xử lý linh hoạt hơn.
- map(lambda row: (row[self.predictCol], row['features'])):
 - o Khóa: row[self.predictCol] - chỉ số của cụm mà điểm dữ liệu được gán vào (ví dụ: cụm 0, 1, 2...).
 - o Giá trị: row['features'] - vector đặc trưng của điểm dữ liệu.
- groupByKey(): Nhóm các điểm dữ liệu lại theo cụm:
 - o Các điểm có cùng giá trị self.predictCol (tức cùng cụm) được nhóm lại với nhau.
 - o Kết quả là một RDD với khóa là số cụm và giá trị là danh sách các vector đặc trưng trong cụm đó:
- mapValues(...): Tính tâm cụm mới (centroid):
 - o features: Danh sách các vector đặc trưng trong mỗi cụm.
 - o zip(*features): Ghép các giá trị của từng chiều đặc trưng lại.
- sum(dimension) / len(features): Tính giá trị trung bình của từng chiều

```
# Convert cluster averages to list
```

```
new_centroids = self.centroids[:]
for cluster, avg_features in cluster_averages.collect():
    new_centroids[cluster] = avg_features
```

- new_centroids: Sao chép danh sách các tâm cụm hiện tại, để tránh thay đổi trực tiếp dữ liệu gốc.
- for cluster, avg_features in cluster_averages.collect(): Duyệt qua từng cụm. Với mỗi cặp (cluster, avg_features) trong danh sách thu thập được. Ở đây

collect() chỉ thu thập kết quả tính toán trung bình cụm (không phải dữ liệu gốc), đồng thời số lượng cụm (k) thường nhỏ, nên dữ liệu thu thập đủ nhỏ để xử lý.

- new_centroids[cluster] = avg_features: Cập nhật tâm cụm với giá trị trung bình vừa tính.

```
# Check if centroids have stabilized
max_centroid_shift = max(
    calculate_distance(self.centroids[i], new_centroids[i]) for i in
range(self.k)
)

# Stop if centroids have minimal movement
if max_centroid_shift < 0.0001:
    break

# Update centroids
self.centroids = new_centroids
```

- calculate_distance(...): Tính khoảng cách Euclid giữa các tâm cụm cũ và mới.
- max_centroid_shift: Giá trị thay đổi lớn nhất giữa các tâm cụm.
- Dừng vòng lặp: Nếu thay đổi nhỏ hơn 0.0001, coi như hội tụ. Nếu không: cập nhật self.centroids với các tâm cụm mới.

```
def transform(self, dataframe, feature_columns):
    df = dataframe.withColumn('features', F.array(feature_columns))
```

```

assign_cluster_udf = F.udf(self.find_closest_cluster, IntegerType())

return
df.withColumn(self.predictCol,assign_cluster_udf(F.col('features'))))

```

- Hàm này áp dụng model K-Means đã huấn luyện để dự đoán cụm cho tập dữ liệu mới.
- Tạo cột features: Tương tự như hàm fit, gộp các cột đặc trưng (feature_columns) thành một mảng.
- Tạo cột features: Tương tự như hàm fit, gộp các cột đặc trưng (feature_columns) thành một mảng.
- Thêm cột dự đoán: Tạo cột mới (self.predictCol, mặc định là prediction) chứa chỉ số của cụm mà mỗi điểm được gán vào.

4.2. DBScan

4.2.1. Cơ sở lý thuyết

- DBSCAN (Density-Based Spatial Clustering of Applications with Noise) là một thuật toán nổi bật được giới thiệu bởi Ester, Kriegel, Sander, và Xu vào năm 1996. DBSCAN dựa trên khái niệm “phân cụm dựa trên mật độ”. Phương pháp này xác định cụm dữ liệu là các khu vực có mật độ cao được tách biệt bởi các khu vực có mật độ thấp. Thuật toán đặc biệt có khả năng xử lý nhiễu và phát hiện cụm dữ liệu với hình dạng phức tạp.
- Các định nghĩa: [1] [2] [3] [4]
 - Eps-neighborhood: vùng lân cận của một điểm dữ liệu P là tập hợp tất cả các điểm dữ liệu nằm trong phạm vi bán kính epsilon (ϵ) xung quanh điểm P. Kí hiệu của tập hợp điểm này là:

$$N_{eps}(P) = \{Q \in D : d(P, Q) \leq \epsilon\}$$

- Trong đó:
 - D là tập hợp tất cả các điểm dữ liệu của tập huấn luyện.
 - $d(P, Q)$ là khoảng cách giữa điểm P và Q.

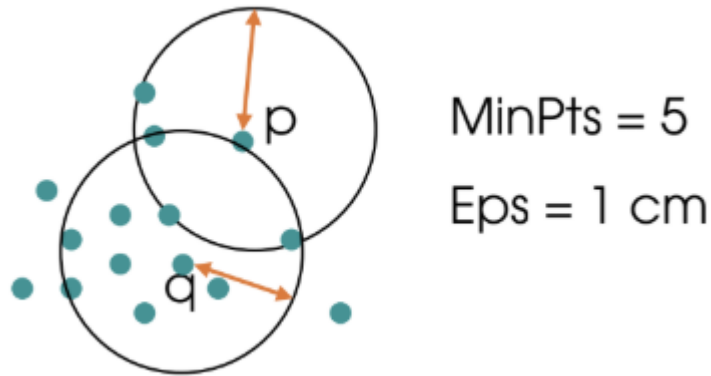


Figure 54. Eps-neighborhood

- Khả năng tiếp cận trực tiếp mật độ (directly density-reachable): dùng để xác định xem một điểm dữ liệu có thể được kết nối với các điểm dữ liệu nằm trong vùng lân cận epsilon hay không. Điểm P được coi là có thể tiếp cận trực tiếp tới điểm Q (cùng với tham số epsilon và minPts) nếu thỏa mãn điều kiện sau:
 - Q nằm trong vùng lân cận epsilon: $Q \in N_{eps}(P)$
 - Số lượng các điểm dữ liệu nằm trong vùng lân cận epsilon tối thiểu là minPts:

$$|N_{eps}(P)| \geq minPts$$

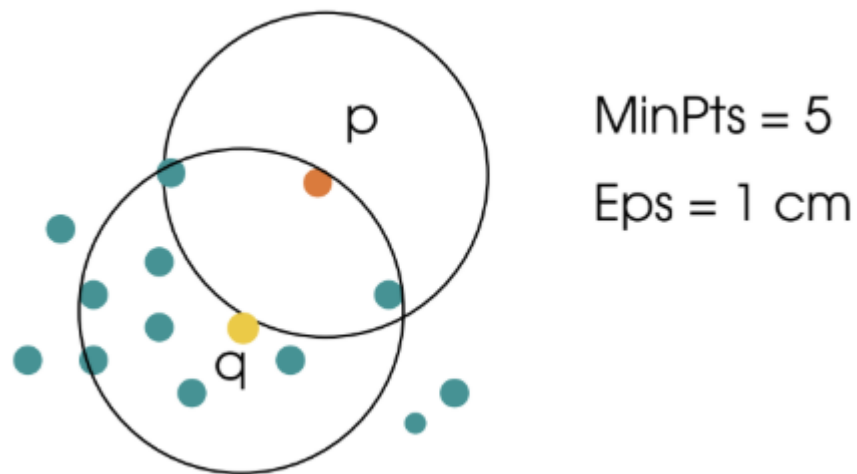
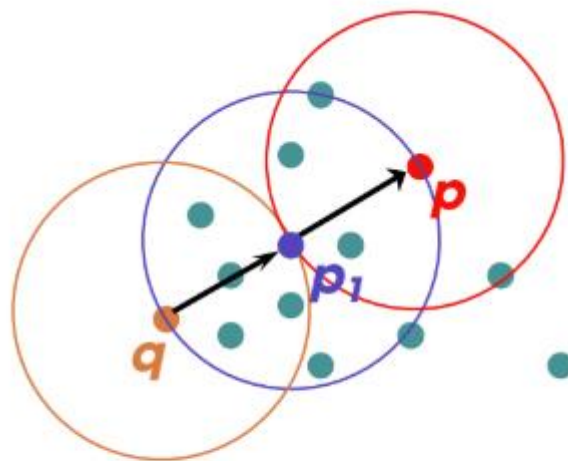


Figure 55. Directly density-reachable

- Khả năng tiếp cận mật độ(density-reachable): liên quan đến hình thành một chuỗi liên kết các điểm trong cụm. Điểm p có khả năng tiếp cận mật độ bởi điểm q nếu dựa trên tham số ϵ và minPts , có một chuỗi các điểm p_1, \dots, p_n với $p_1 = q$ và $p_n = p$ mà p_{i+1} có khả năng tiếp cận trực tiếp mật độ (directly density-reachable) từ p_i .

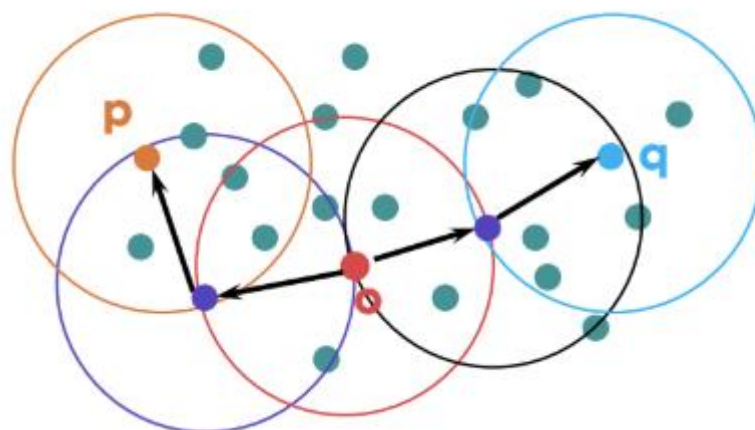


MinPts = 5

Eps = 1 cm

Figure 56. Density-reachable

- Kết nối mật độ(density-connected): một điểm p là kết nối mật độ(density-connected) đến 1 điểm q khi và chỉ khi có một điểm o mà cả hai điểm p và q đều có khả năng tiếp cận mật độ(density-reachable) từ o. Một cụm là một tập các điểm density-connected lớn nhất. Nhiều là điểm không thuộc cụm nào.



MinPts = 5

Eps = 1 cm

Figure 57. Density-connected

- Cụm(cluster): Gọi C_1, C_2, \dots, C_k là một tập con không rỗng của tập dữ liệu D thỏa mãn:
 - $\forall p, q$: nếu $p \in C$ và q có khả năng tiếp cận mật độ(density-reachable) từ p thì $q \in C$.
 - $\forall p, q \in C$: p kết nối mật độ(density-connected) từ q .
- Điểm lõi (core point): một điểm được coi là điểm lõi nếu ít nhất minPts điểm nằm trong bán kính epsilon.
- Điểm biên (border point): các điểm không phải là điểm lõi nhưng nằm trong bán kính epsilon của một điểm lõi.
- Điểm nhiễu (noise point): các điểm không phải là điểm lõi hoặc điểm biên.

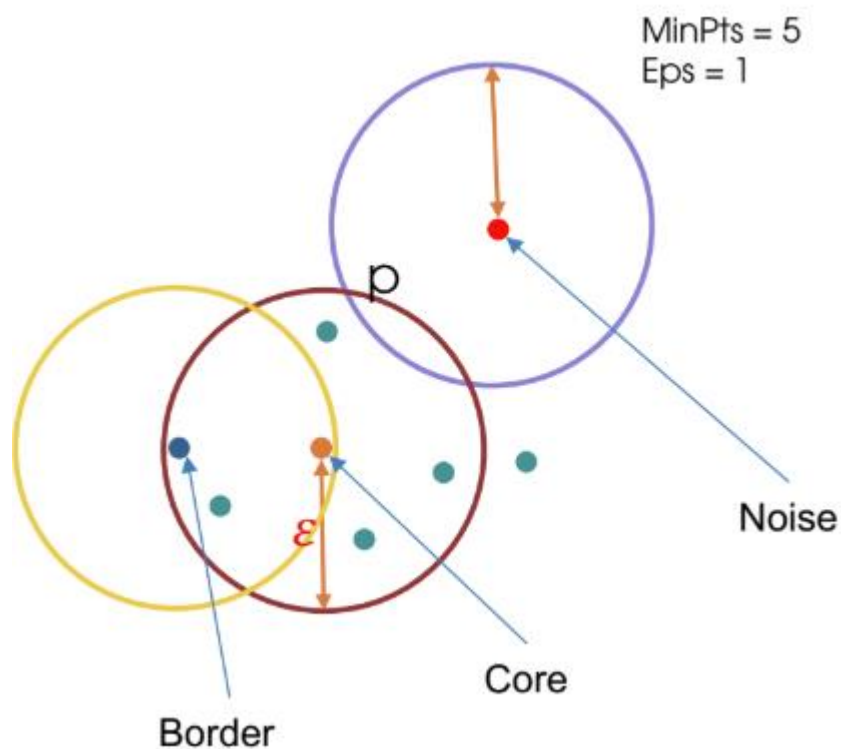


Figure 58. Phân loại các điểm trong DBScan

- Tham số:

- epsilon ($\epsilon > 0$): bán kính lớn nhất vùng lân cận.
- minPts(ngưỡng): số lượng tối thiểu điểm lân cận (bao gồm chính nó) cần thiết để một điểm trở thành core point (điểm lõi).

```

1. DBSCAN(DB, distFunc, eps, minPts) {
    C := 0 // Bộ đếm cụm
    for each point P in database DB {
        if label(P) ≠ undefined then continue // Đã được xử lý trước đó
        Neighbors N := RangeQuery(DB, distFunc, P, eps) // Tìm hàng xóm
        if |N| < minPts then {
            label(P) := Noise // Gán nhãn là Nhiễu
            continue
        }
        C := C + 1 // Nhãn cụm tiếp theo
        label(P) := C // Gán nhãn cho điểm ban đầu
        ExpansionSet S := N \ {P} // Tập hợp các điểm lân cận để mở rộng
        for each point Q in S {
            if label(Q) = Noise then label(Q) := C // Thay đổi Nhiễu thành điểm biên
            if label(Q) ≠ undefined then continue // Đã được xử lý trước đó
            label(Q) := C // Gán nhãn cho hàng xóm
            Neighbors N := RangeQuery(DB, distFunc, Q, eps) // Tìm hàng xóm
            if |N| ≥ minPts then {
                ExpansionSet := ExpansionSet ∪ N // Thêm hàng xóm mới vào tập hợp mở
            }
        }
    }
}

```

4.2.2. Thực nghiệm

- Các bước thực hiện:
 - Bước 1: Xử lý dữ liệu đầu vào: Chọn những cột cần tính toán, loại bỏ các dòng có giá trị null, đánh số thứ tự.
 - Bước 2: Tính khoảng cách Euclidean giữa các điểm dữ liệu, bao gồm

chính điểm đó.

- Bước 3: Khai báo bán kính lân cận eps và số điểm lân cận nhỏ nhất minPts, broadcast hai giá trị này.
- Bước 4: Tìm hàng xóm cho tất cả các điểm. Hàng xóm của một điểm cũng bao gồm chính điểm đó.
- Bước 5: Phân loại các điểm core, border, noise dựa trên số lượng hàng xóm và minPts, khởi tạo cluster id ban đầu cho các điểm là 0.
- Bước 6: Khởi tạo 1 biến cluster_id = 0. Chọn một điểm core từ Dataframe. Lấy các điểm hàng xóm từ điểm core đã chọn, tăng biến cluster_id lên 1, cập nhật cluster_id cho các điểm này bao gồm điểm core đã chọn.
- Bước 7: Từ điểm core point, mở rộng cụm dựa vào hàng xóm của các điểm đã cập nhật cluster id ở trên. Chỉ thêm các điểm core và border vào cụm. Dừng khi không tìm thấy điểm core và border trong các hàng xóm.
- Bước 8: Lặp lại bước 6 đến khi không tìm thấy điểm core trong Dataframe.

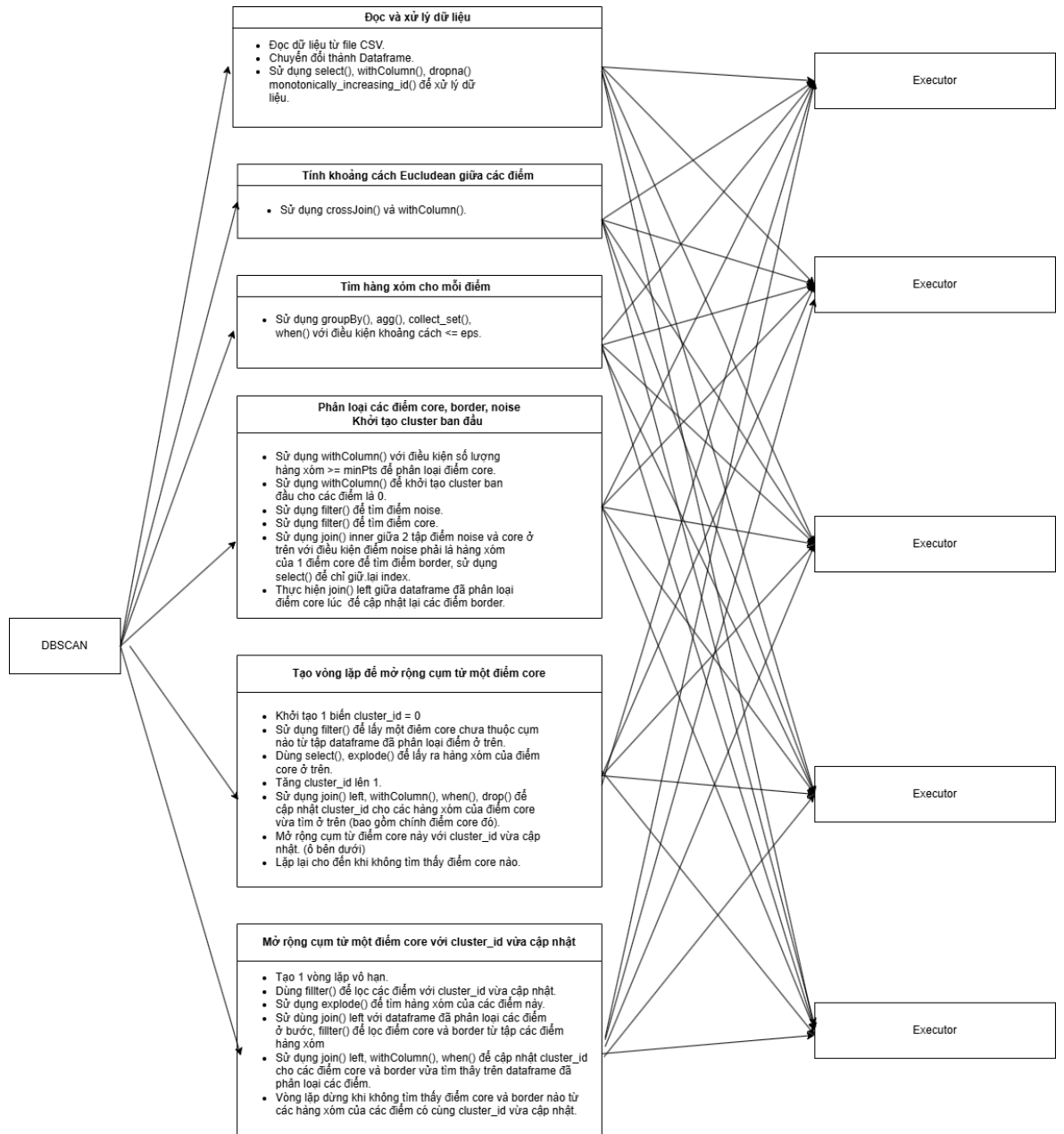


Figure 59. Mô tả song song hóa giải thuật DBScan

- Chạy thuật toán:

- Bước 1: Xử lý dữ liệu đầu vào: Chọn những cột cần tính toán, loại bỏ các dòng có giá trị null, đánh số thứ tự.

```
from pyspark.sql.functions import monotonically_increasing_id
from pyspark.sql import functions as f

df = spark.read.csv("/content/data.csv", header=True, inferSchema=True)
df = df.select(["Read Speed", "Write speed", "Digital Storage Capacity (GB)", "Memory Storage Capacity (GB)"]).dropna()
df = df.withColumn("index", monotonically_increasing_id())
df.show()
```

Read Speed	Write speed	Digital Storage Capacity (GB)	Memory Storage Capacity (GB)	index
0.0	0.0	0.0	0.001736111	0
0.001276596	0.0	0.0	1.09E-4	1
0.0	0.0	5.0E-4	0.020833333	2
0.00212766	0.0	0.0	0.003472222	3
0.0	0.0	0.003255208	0.138888889	4
0.056737589	0.056737589	3.33E-4	0.0	5
0.035460993	0.035460993	1.67E-4	0.0	6
0.0	0.0	0.001	0.0	7
0.0	0.0	8.14E-5	0.003390842	8
0.0	0.0	0.0	4.34E-4	9
0.002836879	0.002836879	0.0	8.68E-4	10
0.0	0.0	9.77E-5	0.0	11
0.030496454	0.030496454	8.14E-5	0.003390842	12
0.0	0.0	4.88E-4	0.020833333	13
0.0	0.0	3.33E-4	0.013888889	14
0.002836879	0.002836879	0.0	4.34E-4	15
0.0	0.0	0.0	1.09E-4	16
0.0	0.0	6.67E-4	0.027777778	17
0.0	0.0	0.0	0.055555556	18
0.0	0.0	8.14E-5	0.0	19

only showing top 20 rows

Figure 60. Xử lý dữ liệu đầu vào

- Bước 2: Tính khoảng cách Euclidean giữa các điểm dữ liệu, bao gồm chính điểm đó.

```
# Dùng cross join để tính khoảng cách một điểm đến các điểm còn lại
df1 = df.alias("df1")
df2 = df.alias("df2")

distance_df = df1.crossJoin(df2) \
    .withColumn(
        "distance",
        f.sqrt(
            (f.col("df1.Read speed") - f.col("df2.Read speed"))**2 +
            (f.col("df1.Write speed") - f.col("df2.Write speed"))**2 +
            (f.col("df1.Digital Storage Capacity (GB)") - f.col("df2.Digital Storage Capacity (GB)")**2 +
            (f.col("df1.Memory Storage Capacity (GB)") - f.col("df2.Memory Storage Capacity (GB)")**2
        )
    )

distance_df.show()
```

te speed	Digital Storage Capacity (GB)	Memory Storage Capacity (GB)	index	Read Speed	Write speed	Digital Storage Capacity (GB)	Memory Storage Capacity (GB)	index	distance
0.0	0.0	0.001736111	0	0.0	0.0	0.0	0.001736111	0	0.0
0.0	0.0	0.001736111	0	0.001276596	0.0	0.0	1.09E-4	1	0.002068136251202275
0.0	0.0	0.001736111	0	0.0	0.0	5.0E-4	0.020833333	2	0.01910376633298886
0.0	0.0	0.001736111	0	0.00212766	0.0	0.0	0.003472222	3	0.002746091491542...
0.0	0.0	0.001736111	0	0.0	0.0	0.003255208	0.138888889	4	0.13719140239913194
0.0	0.0	0.001736111	0	0.056737589	0.056737589	3.35E-4	0.0	5	0.08025853837113089
0.0	0.0	0.001736111	0	0.035460993	0.035460993	1.67E-4	0.0	6	0.050179737140567206
0.0	0.0	0.001736111	0	0.0	0.0	0.001	0.0	7	0.002003517258303...
0.0	0.0	0.001736111	0	0.0	0.0	8.14E-5	0.003390842	8	0.001656731916261952
0.0	0.0	0.001736111	0	0.0	0.0	4.34E-4	0.0	9	0.001302110000000...
0.0	0.0	0.001736111	0	0.002836879	0.002836879	8.68E-4	10	0.004104799828201493	

Figure 61. Tính khoảng cách Euclidean giữa các điểm

- Bước 3: Khai báo bán kính lân cận eps và số điểm lân cận nhỏ nhất minPts, broadcast 2 giá trị này.

```
# Khai báo eps và minPts
eps = 0.2
minPts = 10
eps_broadcast = spark.sparkContext.broadcast(eps)
minPts_broadcast = spark.sparkContext.broadcast(minPts)
```

Figure 62. Khởi tạo eps và minPts và broadcast giá trị

- Bước 4: Tìm hàng xóm cho tất cả các điểm. Hàng xóm của một điểm cũng bao gồm chính điểm đó.

```
# Tính hàng xóm của mỗi điểm
neighbors_df = distance_df.groupBy("df1.index")\
    .agg(f.collect_set(f.when(f.col("distance") <= eps_broadcast.value, f.col("df2.index")))\
        .alias("neighbors"))
neighbors_df.show()
```

index	neighbors
0	[0, 356, 843, 437...]
1	[0, 356, 843, 437...]
5	[0, 356, 843, 437...]
6	[0, 356, 843, 437...]
7	[0, 356, 843, 437...]

Figure 63. Tìm hàng xóm cho tất cả các điểm

- Bước 5: Phân loại các điểm core, border, noise dựa trên số lượng hàng xóm và minPts, khởi tạo cluster id ban đầu cho các điểm là 0.

```
# Đánh dấu điểm core và khởi tạo cluster ban đầu là 0
neighbors_df = neighbors_df.withColumn("isCore", f.size(f.col("neighbors")) >= minPts_broadcast.value)\
    .withColumn("cluster_id", f.lit(0))
neighbors_df.show()
```

index	neighbors	isCore	cluster_id
0	[0, 356, 843, 437...]	true	0
1	[0, 356, 843, 437...]	true	0
5	[0, 356, 843, 437...]	true	0
6	[0, 356, 843, 437...]	true	0
7	[0, 356, 843, 437...]	true	0

Figure 64. Phân loại điểm core và khởi tạo cluster ban đầu cho các điểm

```
# Tìm các điểm non_core_points
non_core_points = neighbors_df.filter(f.col("isCore") == False).select("index").withColumnRenamed("index", "non_core_index")
non_core_points.show()

+-----+
|non_core_index|
+-----+
|          1121|
|          1642|
|          1785|
|          1790|
|          1382|
+-----+
```

Figure 65. Tìm các điểm noise

```
# Tìm các điểm core_points
core_points = neighbors_df.filter(f.col("isCore") == True).select("index", "neighbors").withColumnRenamed("index", "core_index")
result_df = core_points.count()
result_df

1823
```

Figure 66. Tìm các điểm core

```
# Tìm các điểm border_points bằng cách inner join non_core_points và core_points với điều kiện non_core_point xuất hiện trong neighbors
# của core_points
border_points = non_core_points.join(
    core_points,
    f.array_contains(f.col("neighbors"), f.col("non_core_index")), # Điều kiện neighbors chứa non-core point
    "inner" # Giữ lại các non-core points thỏa mãn điều kiện
).select(f.col("non_core_index").alias("index")) # Chỉ cần cột index

# Đánh dấu isBorder = True cho các border points
border_points = border_points.withColumn("isBorder", f.lit(True))
border_points.show()

+-----+-----+
|index|isBorder|
+-----+-----+
+-----+-----+
```

Figure 67. Tìm các điểm border

```
# Thực hiện neighbors left join border_pointst để thêm cột các điểm border vừa tính ở trên
neighbors_df = neighbors_df.join(
    border_points,
    "index",
    "left" # Giữ lại tất cả các điểm trong neighbors_df
).withColumn(
    "isBorder",
    f.when(f.col("isBorder").isNull(), False).otherwise(f.col("isBorder")) # Mặc định False nếu không phải border
)
neighbors_df.show()

+-----+-----+-----+-----+-----+
|index|neighbors|isCore|cluster_id|isBorder|
+-----+-----+-----+-----+-----+
|  0|[0, 356, 843, 437...]| true|      0| false|
|  1|[0, 356, 843, 437...]| true|      0| false|
|  5|[0, 356, 843, 437...]| true|      0| false|
|  6|[0, 356, 843, 437...]| true|      0| false|
|  7|[0, 356, 843, 437...]| true|      0| false|
```

Figure 68. Cập nhật các điểm border vào dataframe

- Bước 6: Khởi tạo 1 biến `cluster_id = 0`. Chọn một điểm core từ Dataframe. Lấy các điểm hàng xóm từ điểm core đã chọn, tăng biến `cluster_id` lên 1, cập nhật `cluster_id` cho các điểm này bao gồm điểm core đã chọn.

```
cluster_id = 0

while not neighbors_df.filter((f.col("isCore")) & (f.col("cluster_id") == 0)).isEmpty():
    # Lấy điểm core đầu tiên chưa thuộc cụm nào
    core_point = neighbors_df.filter((f.col("isCore")) & (f.col("cluster_id") == 0)).limit(1)

    neighbors = core_point.select(f.explode(f.col("neighbors")).alias("neighbor_index")).distinct()

    # Gán cluster_id cho điểm core này, sử dụng neighbors_df left join neighbors để cập nhật cluster_id
    cluster_id += 1
    neighbors_df = neighbors_df.alias("df1").join(
        neighbors.alias("neighbors"),
        f.col("df1.index") == f.col("neighbors.neighbor_index"),
        "left"
    ).withColumn(
        "cluster_id",
        f.when(f.col("neighbors.neighbor_index").isNotNull(), cluster_id).otherwise(f.col("df1.cluster_id"))
    ).drop("neighbor_index")

    # Mở rộng cụm bằng cách gọi expand_cluster
    neighbors_df = expand_cluster(neighbors_df, cluster_id)
```

Figure 69. Mở rộng cụm từ một điểm core

- Bước 7: Từ điểm core point, mở rộng cụm dựa vào hàng xóm của các điểm đã cập nhật `cluster_id` ở trên. Chỉ thêm các điểm core và border vào cụm. Dừng khi không tìm thấy điểm core và border trong các hàng xóm.


```

def expand_cluster(df, cluster_id):
    while True:
        # Lấy các điểm thuộc cụm hiện tại
        cluster_points = df.filter(f.col("cluster_id") == cluster_id)

        # Lấy danh sách hàng xóm của các điểm trong cụm
        neighbors = cluster_points.select(f.explode(f.col("neighbors")).alias("neighbor_index")).distinct()

        # Lọc các điểm mới, chỉ giữ lại các điểm mới bằng left anti join
        new_points = df.join(neighbors, f.col("index") == f.col("neighbor_index"), "left_anti") \
            .filter(f.col("cluster_id") == 0) \
            .select("index", "isCore", "isBorder").distinct()

        # Tách các điểm lõi và biên
        core_points = new_points.filter(f.col("isCore") == True)
        border_points = new_points.filter(f.col("isBorder") == True)

        # Kiểm tra nếu không có điểm mới
        if is_empty(core_points) and is_empty(border_points):
            break

        # Gán cluster_id cho cả core points và border points
        df = df.join(core_points.select("index").withColumnRenamed("index", "core_index"),
                    f.col("index") == f.col("core_index"), "left") \
            .join(border_points.select("index").withColumnRenamed("index", "border_index"),
                    f.col("index") == f.col("border_index"), "left") \
            .withColumn("cluster_id",
                        f.when(f.col("core_index").isNotNull(), cluster_id)
                        .when(f.col("border_index").isNotNull(), cluster_id)
                        .otherwise(f.col("cluster_id"))) \
            .drop("core_index", "border_index")

        # Đảm bảo lưu trữ ở executor cluster, tránh tính lại nhiều lần
        df = df.cache()
    return df

```

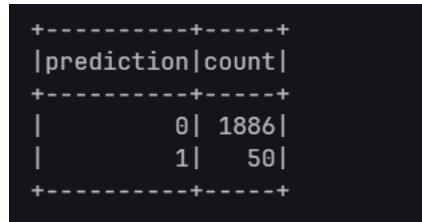
Figure 70. Hàm mở rộng cụm

- Bước 8: Lặp lại bước 6 đến khi không tìm thấy điểm core trong Dataframe.

CHƯƠNG 5: KẾT QUẢ ĐẠT ĐƯỢC

5.1. Kết quả

5.1.1. K-Means



```
+-----+-----+
|prediction|count|
+-----+-----+
|         0| 1886|
|         1|   50|
+-----+-----+
```

Figure 71 Số lượng dữ liệu ở mỗi cụm

- Nhận xét:
 - Dữ liệu có sự chênh lệch lớn giữa 2 cụm. Cụm lớn chứa đa số dữ liệu trong khi cụm nhỏ chỉ chiếm một phần rất nhỏ (khoảng 3%)
 - Cụm lớn:
 - Chứa phần lớn dữ liệu, có thể đại diện cho nhóm chính trong tập dữ liệu.
 - Các điểm trong cụm này có thể có nhiều đặc điểm tương đồng dẫn đến việc gom vào cùng một cụm.
 - Cụm nhỏ:
 - Có thể chứa các **dữ liệu ngoại lệ** hoặc nhóm dữ liệu đặc biệt với các đặc trưng khác biệt rõ rệt so với cụm lớn.

5.1.2. DBScan

```
# Xem kết quả cuối cùng
neighbors_df.groupby("cluster_id").count().show()
```

```
+-----+-----+
|cluster_id|count|
+-----+-----+
|          1| 1787|
|          2|   36|
|          0|    5|
+-----+-----+
```

Figure 72. Kết quả phân cụm bằng DBScan

- Nhận xét:

- Cụm ID = 1:
 - Đặc điểm: Đây là cụm lớn nhất, chiếm tới 97.76% tổng dữ liệu.
 - Ý nghĩa: Cụm này có thể đại diện cho các sản phẩm phổ biến và tiêu chuẩn trên Amazon. Các sản phẩm trong cụm này có thông số kỹ thuật tương tự nhau, thể hiện tính đồng nhất và phổ biến của các mặt hàng.
- ⇒ Đây là nhóm sản phẩm trọng tâm và mang tính chính thống, phản ánh phần lớn thị trường.
- Cụm ID = 2:
 - Đặc điểm: Cụm này nhỏ hơn nhiều so với ID = 1, chỉ chiếm 1.97% tổng dữ liệu
 - Ý nghĩa: Cụm này có thể đại diện cho các sản phẩm có thông số kỹ thuật đặc biệt hoặc khác biệt so với cụm chính. Có thể đây là các sản phẩm ngách, cao cấp, hoặc chuyên biệt, ít phổ biến hơn nhưng có đặc tính nổi trội.
- ⇒ Cho thấy sự hiếm hoi hoặc độc đáo của các sản phẩm này trên thị trường
- Cụm ID = 0:

- Đặc điểm: Đây là các điểm nhiễu, chưa phân cụm, chiếm một phần rất nhỏ của dữ liệu (0.27%)
- ⇒ Đây là các sản phẩm ngoại lệ hoặc dữ liệu sai lệch hoặc thiếu thông tin về thông số kỹ thuật.

5.2. So sánh, đánh giá

5.2.1. K-Means

- Sử dụng SilhouetteEvaluator tự cài đặt và thư viện máy học ClusteringEvaluator để đánh giá chất lượng cụm
 - Đầu tiên là chỉ số silhouette score bằng việc cài đặt không sử dụng thư viện máy học

```
for k in range(2,11):  
    kmean = K_Mean(k)  
    kmean.fit(df, feature_cols)  
    df_fit = kmean.transform(df, feature_cols)  
    evaluator = SilhouetteScore()  
    silhouette_score = evaluator.evaluate(df_fit)  
    silhouette_list.append(silhouette_score)
```

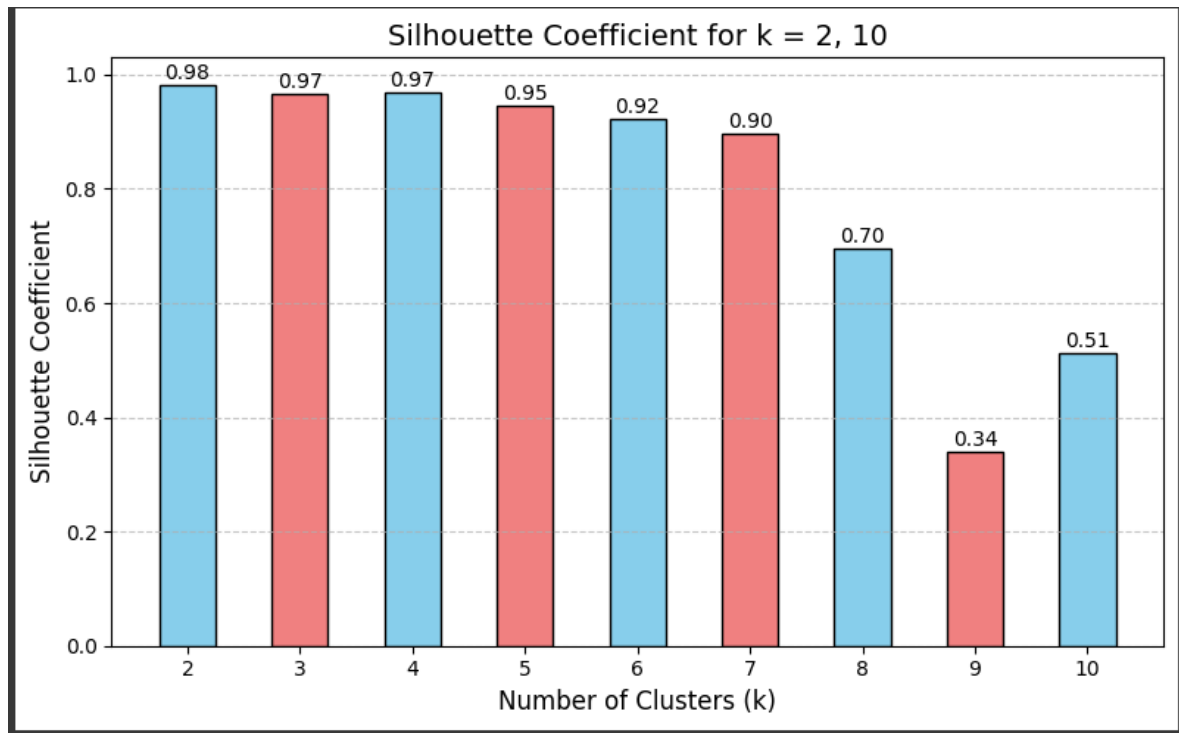


Figure 73 Chỉ số silhouette score cài đặt không dùng thư viện máy học

- Tiếp theo là chỉ số silhouette score với việc sử dụng thư viện máy học

for k in range(2,11):

```
kmeans = KMeans(k=k, seed=1)
```

```
kmeans = kmeans.setFeaturesCol("features")
```

```
model = kmeans.fit(vectorized_df)
```

```
df_result = model.transform(vectorized_df)
```

```
evaluator = ClusteringEvaluator()
```

```
silhouette_score = evaluator.evaluate(df_result)
```

```
silhouette_list.append(silhouette_score)
```

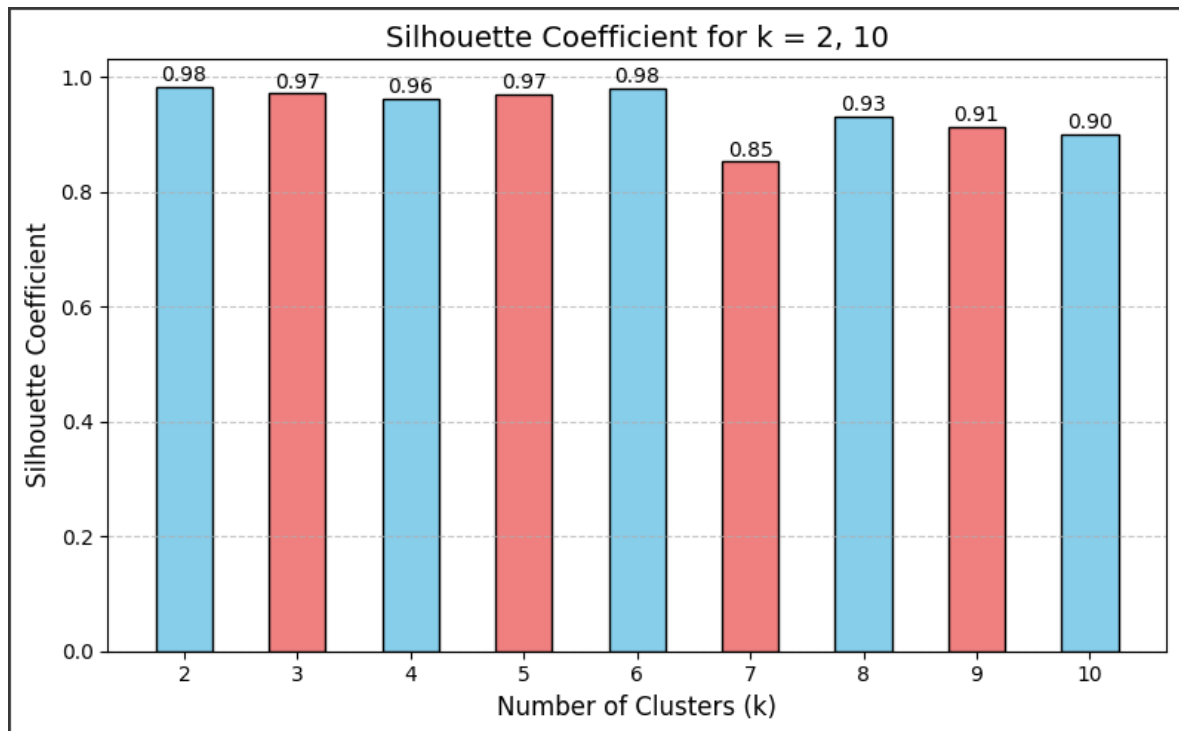


Figure 74 Chỉ số silhouette score cài đặt dùng thư viện máy học

- ❖ So sánh việc phân cụm của việc tự cài đặt và thư viện máy học
 - Dựa vào silhouette score, ta chọn cụm $k=2$

```
[10] feature_cols = ['Read Speed', 'Write Speed','Digital Storage Capacity (GB)','Memory Storage Capacity (GB)']
kmean = K_Mean(k=2)
kmean.fit(df,feature_cols)
df_fit = kmean.transform(df, feature_cols)
evaluator = SilhouetteScore()
silhouette_score = evaluator.evaluate(df_fit)
print(silhouette_score)
```

0.9807285882070823

```
df_fit.show(5)
```

```
df_tmp = df_fit.groupBy('prediction').count().orderBy('prediction')
df_tmp.show()
```

title	price	old_price	rating	reviews	purchases
JSL JDTC 256GB 3...	0.016664844556739288	0.02036855495841464	0.9148936947312738	1.978833307176341...	0.0
USB C Flash Drive...	0.005183342664759051	0.005442602322233656	0.9148936947312738	1.603537335125656E-4	0.01
Avolusion PRO-5U ...	0.014627803835817845	0.014627803835817845	0.0	0.0	0.0
SCICNCE 512GB Fla...	0.009812981020066587	0.009812981020066587	1.0	5.458850502555425E-5	0.0
Apricorn 20TB Aeg...	0.19542443472300838	0.21396706175322222	0.0	0.0	0.0

only showing top 5 rows

prediction	count
0	1886
1	50

Figure 75 Phân cụm không dùng thư viện máy học

```

kmeans = KMeans(k=2, seed=1)
kmeans = kmeans.setFeaturesCol("features")
model = kmeans.fit(vectorized_df)
clustered_df = model.transform(vectorized_df)
clustered_df.show(5)
evaluator = ClusteringEvaluator()
silhouette_score = evaluator.evaluate(clustered_df)
print(silhouette_score)
df_tmp = clustered_df.groupBy('prediction').count().orderBy('prediction')
df_tmp.show(5)

```

```

+-----+-----+-----+-----+
|          title|          price|          old_price|          rating
+-----+-----+-----+-----+
|JSL JDTDC 256GB 3...|0.016664844556739288| 0.02036855495841464|0.914893694731273
|USB C Flash Drive...|0.005183342664759051|0.005442602322233656|0.914893694731273
|Avolusion PRO-5U ...|0.014627803835817845|0.014627803835817845|
|SCICNCE 512GB Fla...|0.009812981020066587|0.009812981020066587|
|Apricorn 20TB Aeg...| 0.19542443472300838| 0.21396706175322222|
+-----+-----+-----+-----+
only showing top 5 rows

0.9813244779824484
+-----+-----+
|prediction|count|
+-----+-----+
|          0| 1887|
|          1|   49|
+-----+-----+

```

Figure 76 Phân cụm sử dụng thư viện máy học

➤ Nhận xét:

- Silhouette score của cả 2 cách cài đặt từ k=2 tới k=7 khá là xêm xêm nhau, nhưng từ k=8 tới k=10 thì lệch đi khá nhiều. Điều đó cho thấy cách sử dụng thư viện làm tốt hơn.
- Bên cạnh đó tốc độ thực thi khi sử dụng thư viện máy học nhanh hơn việc tự cài đặt rất nhiều

5.2.2. DBScan

Silhouette Score = 0.855

(avg_silhouette=0.8554276354987757)

Figure 77. Chỉ số Silhoutte của DBScan

- Nhận xét:
 - Kết quả cho thấy DBSCAN hoạt động rất hiệu quả trên tập dữ liệu này.
 - Các cụm được phân tách tốt, có độ gắn kết cao và ít bị ảnh hưởng bởi điểm nhiễu.
 - ⇒ DBSCAN phù hợp với cấu trúc dữ liệu hiện tại, chứng tỏ được khả năng mạnh mẽ trong việc phân cụm dữ liệu mật độ cao và có nhiễu.

Heatmap

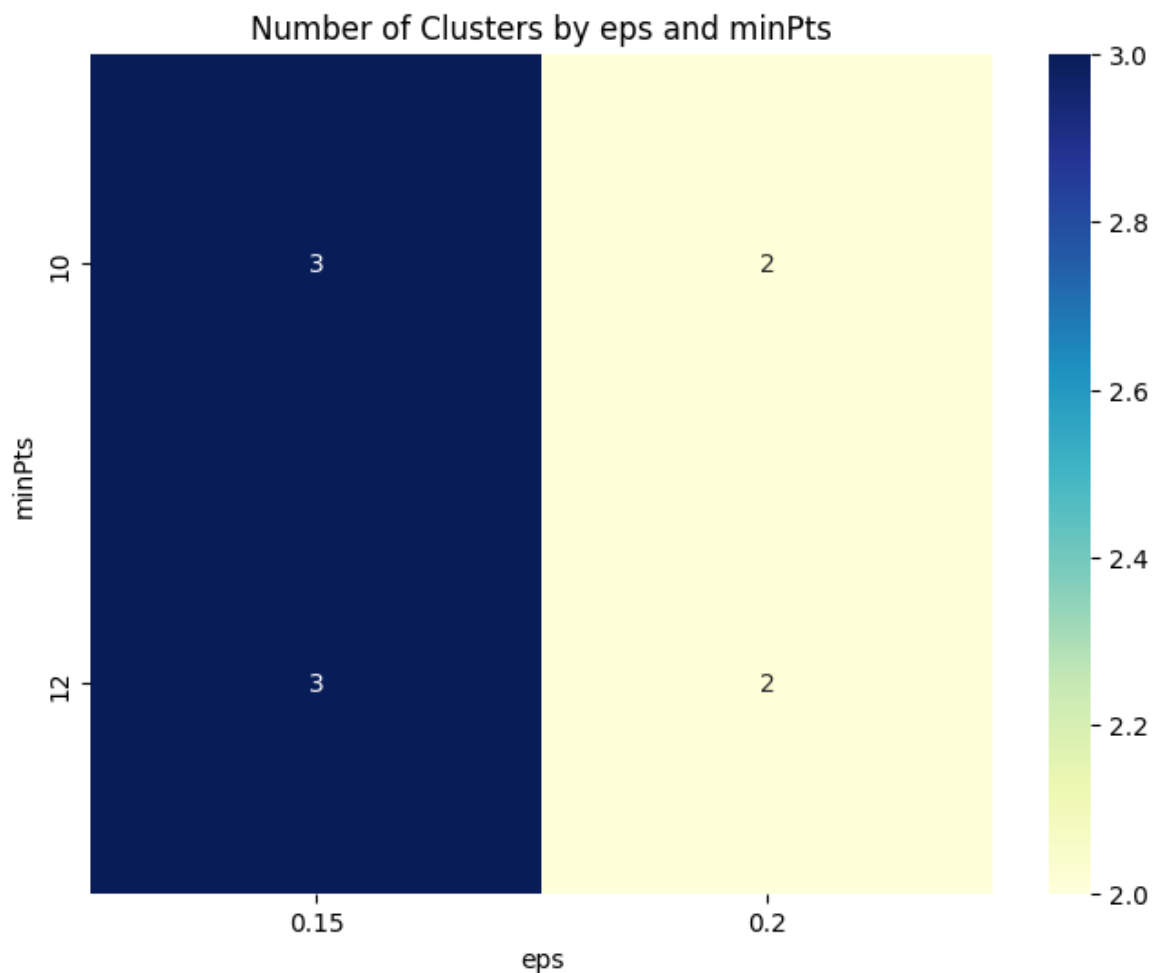


Figure 78. Heatmap của DBScan

- Nhận xét:
 - Ảnh hưởng của epsilon (eps):
 - Khi $\text{eps} = 0.15$, số lượng cụm là 3 (cho cả $\text{minPts} = 10$ và 12).
 - Khi eps tăng lên 0.2 , số lượng cụm giảm xuống còn 2.
 - ⇒ Điều này cho thấy giá trị eps nhỏ hơn giúp phát hiện nhiều cụm hơn, còn eps lớn làm các cụm bị gộp lại.
 - Ảnh hưởng của minPts : Thay đổi minPts từ 10 lên 12 không làm thay đổi số lượng cụm trong cả hai trường hợp $\text{eps} = 0.15$ và $\text{eps} = 0.2$.
 - ⇒ Điều này có nghĩa minPts trong khoảng này chưa ảnh hưởng đáng kể đến kết quả phân cụm.

CHƯƠNG 6: KẾT LUẬN

6.1. Ưu điểm

- Nhận diện được bài toán khai thác dữ liệu lớn và áp dụng được các giải thuật khai thác dữ liệu song song, phân tán.
- Ứng dụng công nghệ học máy hiện đại: Đề tài sử dụng các thuật toán phân cụm tiên tiến như K-means và DBScan đại diện cho ứng dụng thực tiễn của học máy trong việc phân tích dữ liệu của sản phẩm.

6.2. Hạn chế

- Bộ dữ liệu crawl trực tiếp từ trang web của Amazon, nên việc dữ liệu có thể bị thiếu các thuộc tính là không thể tránh khỏi, dẫn tới việc tiền xử lý gặp nhiều khó khăn.
- Nhóm chọn K-means để phân tích dữ liệu, nó phụ thuộc vào việc xác định số lượng cụm k ban đầu và cách xử lý dữ liệu đầu vào, nên còn một số hạn chế về việc cải thiện hiệu suất của thuật toán. Còn về DBScan, thuật toán này cũng yêu cầu 2 tham số là eps và minPts, việc chọn giá trị cho các tham số này không phải lúc nào cũng dễ dàng, kết quả phân cụm có thể không chính xác hoặc không hợp lý. Việc tìm giá trị tối ưu cho các tham số này phụ thuộc vào dữ liệu, và trong một số trường hợp, điều này có thể mất rất nhiều thời gian và công sức.
- Cả 2 thuật toán đều có thể không chính xác nếu dữ liệu có nhiều nhiễu hoặc nếu các cụm không rõ ràng.

6.3. Hướng phát triển

- Thực hiện crawl dữ liệu sản phẩm trên nhiều sàn thương mại điện tử hơn như Tiki, Lazada, Alibaba,... Để có nhiều dữ liệu để phân tích cũng như huấn luyện mô hình.
- Tối ưu hoá hiệu suất, cải thiện tốc độ xử lý của 2 thuật toán K-Means và DBScan.

- Nghiên cứu và áp dụng xử lý dữ liệu theo thời gian thực, đưa ra kết quả dự đoán real-time, hỗ trợ trong mục đích, nhiều đề tài, nhiều chủ đề khác không chỉ dừng ở các sản phẩm được bày bán trên các trang thương mại điện tử.

PHÂN CÔNG CÔNG VIỆC

	Nguyễn Hoàng Đăng Khoa (21520999)	Cù Ngọc Hoàng (21522086)	Nguyễn Trần Gia Kiệt (21522258) (Nhóm trưởng)	Bùi Đình Triệu (21521576)
Tổng quan đề tài	x	x	x	x
Lấy dữ liệu từ Amazon				x
Kỹ thuật tiền xử lý	x			
Kỹ thuật khai thác dữ liệu				
K-Means			x	
DBScan		x		
Kết quả đạt được				
K-Means			x	
DBScan		x		
Kết luận			x	

Làm báo cáo	x	x	x	x
Làm slide	x	x	x	x
Hoàn thành (%)	100%	100%	100%	100%

TÀI LIỆU THAM KHẢO

- [1] P. D. Khanh, “Phương pháp phân cụm dựa trên mật độ (Density-Based Clustering),” Deep AI KhanhBlog, 2021. [Trực tuyến]. Available: https://phamdinhkhanh.github.io/deepai-book/ch_ml/DBSCAN.html.
- [2] Sachinsoni, “Clustering Like a Pro: A Beginner’s Guide to DBSCAN,” Medium, 2023 December 2023. [Trực tuyến].
- [3] D. P. Long, *Tài liệu bài giảng: KHAI THÁC DỮ LIỆU – IS252 - Chương 7: Gom cụm*, Ho Chi Minh: UIT.
- [4] O. Yenigün, “DBSCAN Clustering Algorithm Demystified,” builtin, 11 March 2024. [Trực tuyến]. Available: <https://builtin.com/articles/dbscan>.
- [5] Geeksforgeeks, “K means Clustering – Introduction,” [Trực tuyến]. Available: <https://www.geeksforgeeks.org/k-means-clustering-introduction/>.
- [6] Geeksforgeek, “K-Means Clustering using PySpark Python,” [Trực tuyến]. Available: <https://www.geeksforgeeks.org/k-means-clustering-using-pyspark-python/>.
- [7] E. Ding, “From Scratch: How to Code K-Means in Python (No Sklearn) for Machine Learning Interviews!,” [Trực tuyến]. Available: <https://www.youtube.com/watch?v=uLs-EYUpGAw>.
- [8] A. Tripathi, “Kmeans from Scratch with Silhoutte and elbow curve,” Kaggle, [Trực tuyến]. Available: <https://www.kaggle.com/code/achintyatripathi/kmeans-from-scratch-with-silhoutte-and-elbow-curve>.
- [9] D. P. Long, *Tài liệu bài giảng: KHAI THÁC DỮ LIỆU - IS252 - Chương 2: Tiền xử lý dữ liệu*, Ho Chi Minh: UIT.

- [10] geeksforgeeks, “Data Preprocessing in Data Mining,” [Trực tuyến]. Available: <https://www.geeksforgeeks.org/data-preprocessing-in-data-mining/>.
- [11] Airflow, “Documentation” [Trực tuyến]. Available: <https://airflow.apache.org/docs/>.
- [12] Kmeans, “k-means clustering” [Trực tuyến]. Available: https://en.wikipedia.org/wiki/K-means_clustering.